U.S.N.A. --- Trident Scholar project report; no. 366 (2008)

Targeting Pod Effects on Weapons Release from F-18C Hornet

by

Midshipman 1/C William H. Godiksen, USN
United States Naval Academy
Annapolis, Maryland

_____
(signature)

Certification of Adviser Approval

Assistant Professor Eric N. Hallberg
Aerospace Engineering Department

_____
(signature)

_____
(date)

Acceptance for the Trident Scholar Committee

Professor Joyce E. Shade
Deputy Director of Research & Scholarship

_____
(signature)

_____
(date)

USNA-1531-2

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATE COVERED
	1 May 2006	

4. TITLE AND SUBTITLE

Targeting Pod Effects on Weapons Release from the F-18C Hornet

5. FUNDING NUMBERS

6. AUTHOR(S)

-

Godiksen III, William H.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)		8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
US Naval Academy Annapolis, MD 21402		Trident Scholar project report no. 366 (2006)

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
This document has been approved for public release; its distribution is UNLIMITED.	

13.ABSTRACT. In recent years, the capability of computational fluid dynamics (CFD) to predict the trajectories of stores has greatly expanded. In this study, a CFD code developed at NASA has been used to explain an unexpected flight test result in order to assess the accuracy of CFD as applied to store separation. In December of 1998 during routine bombing practice at Fallon, NV, an F-18C released a Mark 82 JDAM bomb which impacted the Targeting Forward-Looking Infrared Pod (TFLIR) on the aircraft fuselage. This trajectory was entirely unexpected as that flight condition had previously been cleared as safe for store release. It had been assumed that the addition of the TFLIR would not cause a significant change to the aircraft's flowfield; an assumption which proved inaccurate. This study was designed to investigate whether CFD analysis could predict and explain the results of the aforementioned flight, and to determine the efficacy of CFD to predict and explain the effects of three different targeting pods of different geometries all carried by the F-18C Hornet. The three pods examined in this study were the TFLIR, the ATFLIR, and the Litening pod. The TFLIR and ATFLIR are geometrically quite similar while the Litening pod is both longer and wider. Surprisingly, flight tests showed that the ATFLIR and TFLIR pods had significantly different effects on the aerodynamic loads created on the bomb. Initial speculation centered on the physical differences between the forward ends of the pods, but this research revealed that the most important aspect was the shape and placement of the rear end of the pods. The initial investigation analyzed the aerodynamic effects of each pod on a bomb located adjacent to the pod on the inboard pylon beneath the wing.
(cont on p.2)

14. SUBJECT TERMS	15. NUMBER OF PAGES
store separation, computational fluid dynamics, CFD, TFLIR, ATFLIR, F-18	**95**
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

ABSTRACT

In recent years, the capability of computational fluid dynamics (CFD) to predict the trajectories of stores has greatly expanded. In this study, a CFD code developed at NASA has been used to explain an unexpected flight test result in order to assess the accuracy of CFD as applied to store separation.

In December of 1998 during routine bombing practice at Fallon, NV, an F-18C released a Mark 82 JDAM bomb which impacted the Targeting Forward-Looking Infrared Pod (TFLIR) on the aircraft fuselage. This trajectory was entirely unexpected as that flight condition had previously been cleared as safe for store release. It had been assumed that the addition of the TFLIR would not cause a significant change to the aircraft's flowfield; an assumption which proved inaccurate.

This study was designed to investigate whether CFD analysis could predict and explain the results of the aforementioned flight, and to determine the efficacy of CFD to predict and explain the effects of three different targeting pods of different geometries all carried by the F-18C Hornet. The three pods examined in this study were the TFLIR, the ATFLIR, and the Litening pod. The TFLIR and ATFLIR are geometrically quite similar while the Litening pod is both longer and wider. Surprisingly, flight tests showed that the ATFLIR and TFLIR pods had significantly different effects on the aerodynamic loads created on the bomb. Initial speculation centered on the physical differences between the forward ends of the pods, but this research revealed that the most important aspect was the shape and placement of the rear end of the pods.

The initial investigation analyzed the aerodynamic effects of each pod on a bomb located adjacent to the pod on the inboard pylon beneath the wing. The tapered trailing surface of the ATFLIR pod caused a strong shockwave to form when speeds approached the speed of sound, and it was this shockwave crossing the tail fins of the weapon that caused the adverse change in trajectory. CFD analysis revealed that the TFLIR and Litening pods caused weaker shocks off their aft ends. Furthermore, the extended length of the Litening pod caused the aft shock to impinge less on the adjacent store tail fins. Using the results of the CFD analysis, trajectory simulation predictions were accomplished using the Navy Generalized Separation Package

(NAVSEP). Results from these predictions compared favorably with flight test results, confirming the accuracy of the CFD analysis.

This research has demonstrated the ability of computational fluid dynamics to predict store separation characteristics accurately, revealing the potential for significant savings in terms of cost and schedule for future weapons programs. Additionally, this research has provided invaluable insight into the effects of the three targeting pods on the weapon release characteristics of an F-18C carrying a targeting pod, providing data that can be used to make subtle design changes to the aft geometry of the pods in order to create a safer and more combat-effective fighter.

Keywords: store separation, computational fluid dynamics, CFD, TFLIR, ATFLIR, F-18

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Assistant Professor Eric Hallberg, whose expertise, guidance, and enthusiasm were essential to the execution of this project. I would also like to thank Dr. Alexis Cenko of Naval Air Systems Command for his guidance and expert knowledge in the area of research.

Next, I would like to thank Ms. Linda Adlum for her help with the parallel processing necessary for this project.

Finally, I would like to thank Professor Joyce Shade and the Trident Scholar Committee for their constructive input and the opportunity to conduct this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

6DOF ..... six degrees of freedom
ATFLIR ..... Advanced Targeting Forward Looking Infrared Pod
CFD ..... Computational Fluid Dynamics
JDAM ..... Joint Direct Attack Munition
NAVAIR ..... Naval Air Systems Command
NAVSEP ..... Navy Generalized Separation Package

TFLIR ..... Targeting Forward Looking Infrared Pod
C_x ..... axial force coefficient
C_y ..... sideforce coefficient
C_z ..... normal force coefficient
C_l ..... rolling moment coefficient
C_m ..... pitching moment coefficient
C_n ..... yawing moment coefficient
C_p ..... pressure coefficient
α ..... upwash angle, angle of attack
β ..... sidewash angle
ψ ..... yaw angle
θ ..... pitch angle
φ ..... roll angle

1 INTRODUCTION

In order to establish safe flight conditions for the release of bombs or other stores from attack aircraft, the Navy conducts flight tests at various aircraft attitudes, Mach numbers, and store configurations and determines the initial path taken by the falling store. This determination of path is generally made using a series of high speed photographs, known as photogrammetrics, or the analysis of data taken from an accelerometer located on the store itself, known as telemetry. Though very accurate, many such flight tests are necessary in order to approve a range of acceptable flight conditions, and these are costly in both time and money. In the absence of pre-flight analysis, the most benign flight condition is chosen as the starting point of the flight test, typically fully sub-sonic. The release envelope is gradually expanded through subsequent releases by increasing the Mach number and altitude. Many such flights are required to reach the boundary of the aircraft flight envelope.

The number and duration of flights required can be significantly reduced by predicting trajectories before flights are begun. These predictions are made using store separation analysis, which is a branch of aerodynamics concerned with the trajectories of bombs, missiles and fuel tanks immediately after they are detached from the host aircraft. This field is very important as these stores do not necessarily move downward. In the transonic flight regime, aerodynamic forces can dominate the normal gravitational forces causing the store to “fly” in unexpected directions. There have been cases where the store actually moved up or sideways and impacted the releasing aircraft, occasionally causing extensive damage. Store separation analysis can be used to identify these high-risk release conditions prior to encountering them during flight tests. A good rule of thumb is that as the store separates from the aircraft, the miss distance, or closest point between the store and the aircraft, should continually increase. Furthermore, the miss distance should be greater than six inches by 300 microseconds after release. Miss distances less than six inches may be permissible if the release condition is well understood but in no case should the distance be less than two inches.

Clearing aircraft with stores that will have a miss distance of six inches or less requires significant analysis. This pre-flight flow analysis is accomplished in both the wind tunnel and

through computational fluid dynamics (CFD). Prior to any flight testing, predicted trajectories are obtained using one or both of these methods, and these results are used to determine which configurations require flight tests and to what extent. For instance, a clearance to Mach 0.97 may require a build-up approach beginning at a benign flight condition such as Mach 0.88 and progressing up to Mach 0.97 at steps of 0.02 Mach. Extensive wind tunnel and CFD analysis could permit fewer steps in the build-up to the endpoint if CFD and wind tunnel analysis shows the endpoint to be safe, and interim flight test steps match predictions.

1.1 Background

The inspiration for this research began with routine bombing practice conducted in Fallon, Nevada in December of 1998. The pilot was flying an F-18C aircraft with a Targeting Forward-Looking Infrared Pod (TFLIR) mounted on the side of the plane's fuselage and a Mark-82 bomb hanging from the inboard wing pylon adjacent to the targeting pod. Figure 1 below shows an F-18C aircraft with a TFLIR attached and a fuel tank on the inboard pylon.

Figure 1. TFLIR pod mounted on fuselage of F-18C

When the pilot dropped the Mark 82 from his aircraft, the nose of the bomb yawed away from the fuselage and caused the bomb's tail fins to impact the TFLIR. This result was unexpected as this flight condition had been cleared for safe release in the aircraft's tactical

manual. An investigation soon revealed that the TFLIR was considered a part of the aircraft and that its effect on store separation had been assumed to be negligible. As a result, neither wind tunnel nor flight testing had been done to determine what effect it might have. After this incident, the Navy decided to begin a flight test program in order to establish safe release parameters.[1]

At this same time the Navy introduced the Advanced Targeting Forward Looking Infrared Pod (ATFLIR), which is geometrically similar to the TFLIR but significantly more capable. A picture of the ATFLIR pod mounted on an F-18C can be seen below in Figure 2. The main difference in shape between these two pods is the fairing on the leading edge of the ATFLIR, which is not present on the TFLIR. In most other aspects, these pods look essentially identical. There are subtle differences in the geometry of the trailing end of the pods which were initially not thought to be significant compared to the larger differences in their front-end geometries. This research showed this assumption to be wrong.

Figure 2. ATFLIR pod mounted on fuselage of F-18C[2]

[1] Rothback, N., and Cenko, A., "Dangers of Aircraft Modifications Without Conducting an Investigation into the Effects on the Aircraft Flowfield and Flying Qualities," 2001, p. 1.
[2] Rothback, N., and Cenko, A., "Dangers of Aircraft Modifications Without Conducting an Investigation into the Effects on the Aircraft Flowfield and Flying Qualities," 2001, p. 1.

This pod was examined in the flight test program in the same manner as the TFLIR. It was expected that the ATFLIR would have nearly the same effect on the aircraft's flowfield as the TFLIR due to their geometric similarity. However, flight test results soon proved otherwise. At speeds just under the speed of sound, between Mach 0.90 and 0.95, the flight test results showed significant differences in the trajectories of bombs dropped next to the TFLIR versus those beside the ATFLIR. Although the cause of this variance was not understood, time and schedule constraints precluded further research. The flight test program concluded by restricting the release of certain stores in proximity to either targeting pod to a subset of the full combat aircraft flight envelope.

While these test flights were successful in establishing safe store release conditions for these pods, they did not produce a full understanding of the effect of the (A)TFLIR pod on the F-18C flowfield. Furthermore, the full operating envelope of the combat aircraft was restricted. Analysis of this release condition is challenging. The geometric differences between the two pods are subtle and the flowfield at the Mach number of interest is fully transonic with a number of shocks forming and moving as the store is released. Computational Fluid Dynamics is an ideal choice for analysis of this scenario.

Over the past ten years CFD technology has matured to the point where solving this type of problem is not only possible, but practical and cost effective. Computational resources available to store separation engineers at NAVAIR or researchers at USNA have improved to the point where run times of a day or two for a single flight condition are possible. Furthermore, CFD provides methods of flow visualization and analysis not available through wind tunnel testing or flight tests. This research used flow visualization methods available through CFD not only to predict the motion of the store, but also to explain how subtle changes in the aircraft geometry can affect that motion.

1.2 Store Separation Analysis

Until the 1960s store separation analysis consisted solely of rudimentary flight tests. Stores were dropped at gradually increasing speeds until the store came too close to the aircraft

or in many cases hit the aircraft, occasionally destroying it.[3] In the 1960s wind tunnel testing became sophisticated enough to measure the forces and moments on a model of the store in an aircraft's flowfield and quickly became a widely used part of store separation programs. For the following twenty years, these were the two methods used to predict a store's trajectory. Computational Fluid Dynamics became capable of determining the forces and moments on a store in the 1980s, but it wasn't until the middle of the 1990s that this field advanced enough to provide useful data in reasonable amounts of time. At this point CFD became the third leg in the store separation analysis triad, in which each component serves to validate and complement the others. A graphical representation of this integrated process can be seen below in Figure 3.

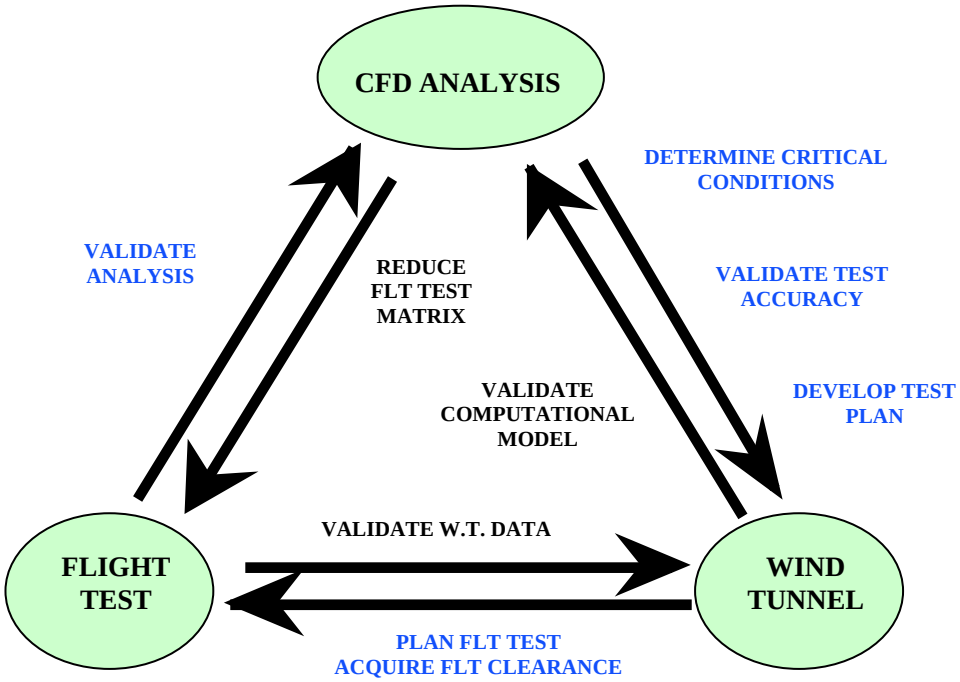


Figure 3. Integrated store separation process

All three of these methods are used in modern store separation analyses. CFD is usually used first to establish dangerous conditions which will require testing in the wind tunnel and to establish the overall wind tunnel test plan. Wind tunnel data is then used to validate the computational model and develop the flight test matrix, seeking to test all possibly dangerous

[3] Cenko, A., "Experience in the Use of Computational Aerodynamics to Predict Store Release Characteristics," *Progress in Aerospace Sciences*, Vol. 37, 2001, p 479.

conditions while minimizing the number of flights required. The results of flight tests then serve to validate both the wind tunnel and CFD data, which can be used to improve both methods. This combination of methods follows the standard approach taken by NAVAIR.[4]

The primary goal of store separation analysis is to predict the initial trajectory of the store as it separates from the aircraft. This trajectory can then be used to find the miss distance of the store, which is undesirable below six inches and should never be below two inches. Using CFD analysis there are multiple ways of predicting store trajectories, which can be grouped into two general categories: the time-accurate approach and the grid method.[5] The grid method was used in this research.

The time-accurate method begins with the store in the carriage position, then applies the ejector force and solves the flow to determine the forces and moments on the store. These forces and moments are then integrated over a time step to find the next position of the store, where the flow is solved again and the store is moved for a second time. This process continues for as long as is necessary. The grid method makes the same fundamental calculations but separates solving the flow and integrating the forces into two separate processes. The flow is solved first. Many different positions and orientations of the store beneath the wing of the aircraft are chosen, and the flow is solved at each of these to obtain the forces and moments on the store at each location. The solutions at each position and orientation create a data grid, giving this method its name. The actual positions and orientations chosen for solving are determined using estimates of where the store will go, and more data points can be added at any time if the behavior is unexpected. This is the most time-intensive aspect of the process since the solution at each location requires a separate run of the flow solver.

The second component of the grid method is a program which solves the rigid-body equations of motion. Known as a 6DOF program because the store is free to move in all three directions and rotate in three others, this program uses the grid data generated by the flow solver

---

[4] Cenko, A., "Experience in the Use of Computational Aerodynamics to Predict Store Release Characteristics," *Progress in Aerospace Sciences*, Vol. 37, 2001, p 479.
[5] Cenko, A., "Experience in the Use of Computational Aerodynamics to Predict Store Release Characteristics," *Progress in Aerospace Sciences*, Vol. 37, 2001, p 481.

as well as characteristic properties of the store itself to calculate the trajectory. Figure 4 below shows a diagram of the trajectory prediction process using the grid method.

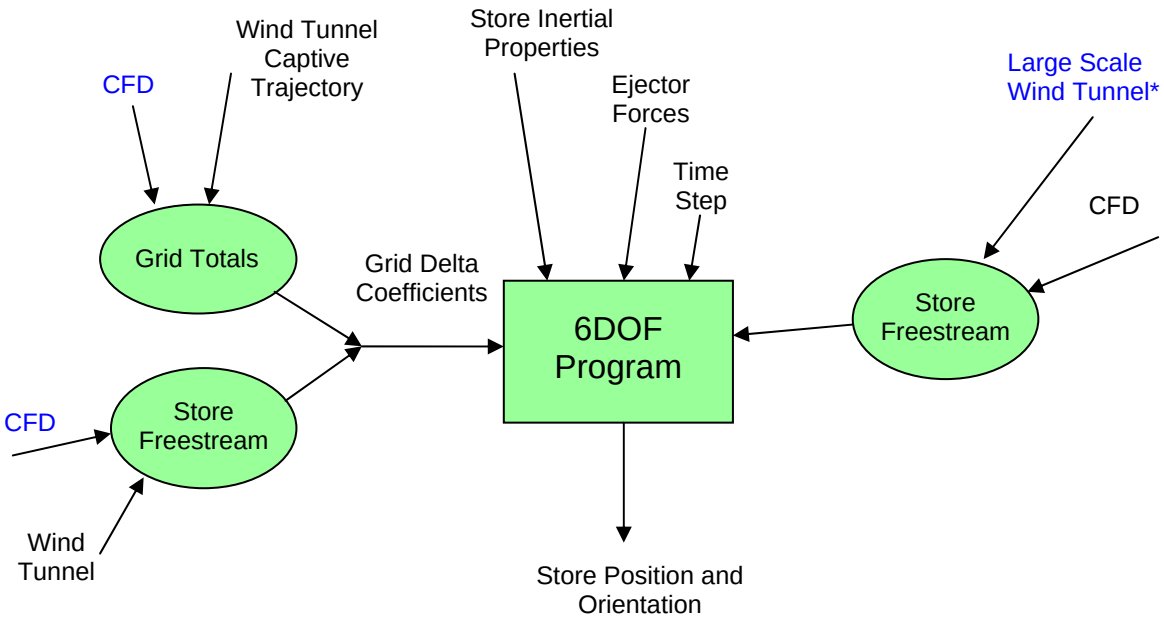


Figure 4. Aspects of trajectory calculation using grid method

As this diagram shows, the same 6DOF program can use wind tunnel or CFD data to perform the trajectory calculations. A well-planned analysis program will actually use data from both wind tunnel and CFD tests to calculate trajectory. While CFD analysis is generally best for large models that need to be substantially scaled down for a wind tunnel, full-scale wind tunnel data of the store by itself, known as freestream data, is considered the most accurate data available. This research project made use of such wind tunnel data generated during previous tests of the store.

The CFD test plan used in this research consisted first of solutions determined in each configuration without a store. The eight Mach numbers chosen for testing were Mach 0.80, 0.85, 0.88, 0.90, 0.92, 0.95, 0.98, and 1.05. An additional five cases were planned for the model of the F-18C aircraft's engine. Validation of the GBU-31 JDAM required another thirteen solutions. Finally, thirteen Mach numbers were chosen to test each configuration with the store in carriage position, including Mach 0.80, 0.85, 0.87, 0.88, 0.89, 0.90, 0.91, 0.92, 0.93, 0.94, 0.95, 0.98, and 1.05. In total, the test plan consisted of 102 test points.

1.3 CFD Development

Computational fluid dynamics is based on three fundamental physical principles given in equation form below. These principles are conservation of mass, conservation of momentum, and conservation of energy.[6] There are multiple equivalent forms of each equation, some more suited to particular aspects of CFD analysis than others. Shown below in Equation 1 through Equation 3 are the governing equations for an unsteady, compressible, viscous flow.

Conservation of mass:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0$$

Equation 1

Conservation of momentum:

$$\frac{\partial (\rho u)}{\partial t} + \nabla \cdot (\rho u \mathbf{V}) = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x$$

$$\frac{\partial (\rho v)}{\partial t} + \nabla \cdot (\rho v \mathbf{V}) = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y$$

$$\frac{\partial (\rho w)}{\partial t} + \nabla \cdot (\rho w \mathbf{V}) = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z$$

Equation 2

Conservation of energy:

$$\frac{\partial}{\partial t}\left[\rho\left(e + \frac{V^2}{2}\right)\right] + \nabla \cdot \left[\rho\left(e + \frac{V^2}{2}\right)\mathbf{V}\right] = \rho \dot{q} + \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)$$
$$- \frac{\partial (up)}{\partial x} - \frac{\partial (vp)}{\partial y} - \frac{\partial (wp)}{\partial z} + \frac{\partial (u\tau_{xx})}{\partial x} + \frac{\partial (u\tau_{yx})}{\partial y} + \frac{\partial (u\tau_{zx})}{\partial z} + \frac{\partial (v\tau_{xy})}{\partial x} + \frac{\partial (v\tau_{yy})}{\partial y} + \frac{\partial (v\tau_{zy})}{\partial z}$$
$$+ \frac{\partial (w\tau_{xz})}{\partial x} + \frac{\partial (w\tau_{yz})}{\partial y} + \frac{\partial (w\tau_{zz})}{\partial z} + \rho \mathbf{f} \cdot \mathbf{V}$$

Equation 3

[6] Anderson, John D., Computational Fluid Dynamics: The Basics with Applications. New York: McGraw-Hill, Inc., 1995, p 38.

This set of equations was first compiled independently by M. Navier and G. Stokes at the beginning of the 19th century, and are thus known as the Navier-Stokes equations.[7] A simpler version of these equations that does not include viscous forces was formulated by Euler and is also frequently used in CFD applications. There is no known analytical solution to these equations, and as such all solutions to the equations are numerical approximations.

Numerical solutions to these equations require discretization, meaning that the flow parameters are computed at a finite number of points in the grid. This requirement necessitates breaking up the volume in which to solve the equations into small pieces. The creation of these small pieces of volume is known as grid generation and is an important part of the CFD process. There are two categories of grid generation: structured and unstructured. Structured grids have an inherent order to them. While the cells do not have to be rectangular, they must be able to be transformed into rectangles, which restricts the flexibility of the grid. In some cases, the cells are created completely rectangular. An example of such a volume grid can be seen below in Figure 5, where the cells shown are part of a plane cut through the grid perpendicular to the aircraft's axis.

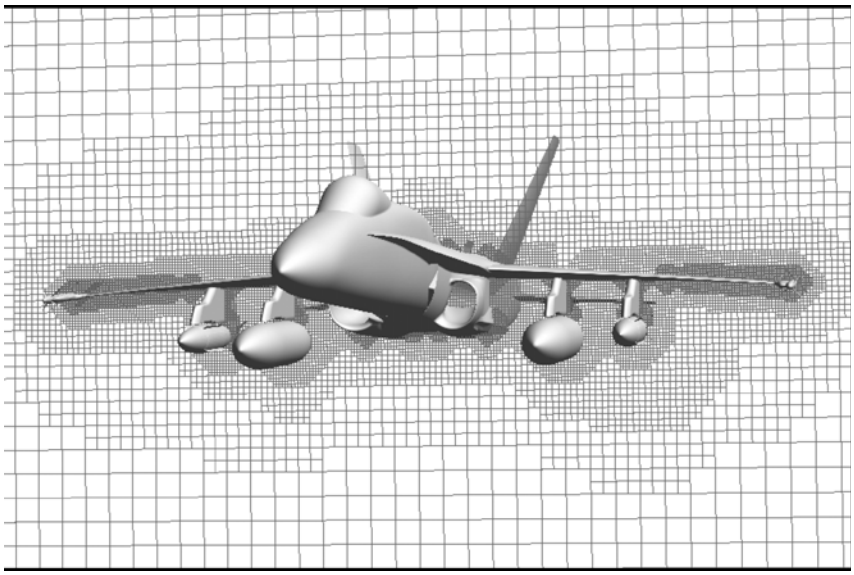


Figure 5. Sample structured volume grid slice

[7] Anderson, John D., Computational Fluid Dynamics: The Basics with Applications. New York: McGraw-Hill, Inc., 1995, p 64.

Structured grids are advantageous in that their order simplifies the calculations and data structure on the computer, but can be poor choices in that they are labor intensive when used to model complex shapes. The CFD engineer has excellent control of the grid sizing everywhere on the model, an important consideration for obtaining an accurate solution. The process of creating the structured grids, however, takes a long time. Unstructured grids, by contrast, are exactly what their name implies. Their lack of inherent structure makes them an ideal choice for complex geometries. Controlling cell size is not done explicitly as in the case of structured grids. There are methods, however, within most CFD packages to adjust cell sizes on the model in an iterative manner. The CFD engineer employs these methods until the surface grid is sufficiently well-defined. Even though the process is iterative, the total time expended in creating an unstructured grid is far less than that of a structured grid. As a result unstructured grids were chosen for the present research. A sample unstructured surface grid can be seen below in Figure 6.

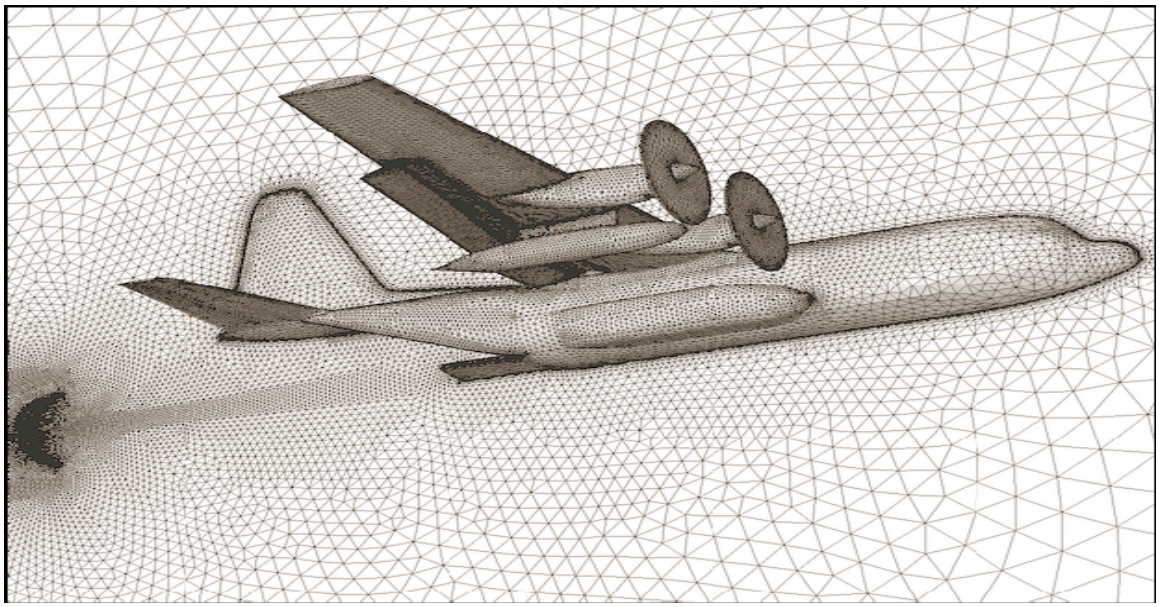


Figure 6. Sample unstructured surface grid[8]

As this figure shows there is no set order applied to each cell. While they are ideally equilateral in shape, they are frequently not equilateral in the vicinity of the aircraft in order to

[8] Frink, Neal, et. al., "TetrUSS," http://aaac.larc.nasa.gov/tsab/tetruss/mac/TetrUSS-2005-0812.pdf. October 2006.

match the surface of the geometry. This flexibility in shape is what makes these grids ideal for complex shapes. Unstructured grids have another inherent benefit in the size of the cells. While cell size in structured grids must vary in a uniform manner as seen in Figure 5, there is no such requirement in unstructured grids. The size of the cells can be specified to obtain high resolution in areas of interest while reducing the total number of cells by coarsening the grid elsewhere.

Once a grid has been created and a flow solver has been run, a "solution" to the flow has been generated. A solution to a flow consists of the values of the flow's primitive variables in each cell of the grid. These variables are density of the air, its velocity in the x, y, and z directions, and the internal energy of the air. Each of these parameters is useful for the analysis of airflow over a given configuration, but the most important parameters to this research are the static pressure and shear forces, which can be obtained from the other quantities found. These two quantities can be integrated over the surface of the store to determine the aerodynamic forces and moments on the store. Finding these forces and moments was the goal of generating flow solutions in this project. Of these two quantities, pressure tends to be the dominant factor and is therefore usually the focus when examining flow solutions. Figure 7 below shows the pressure distribution on the F-18C with a TFLIR attached.

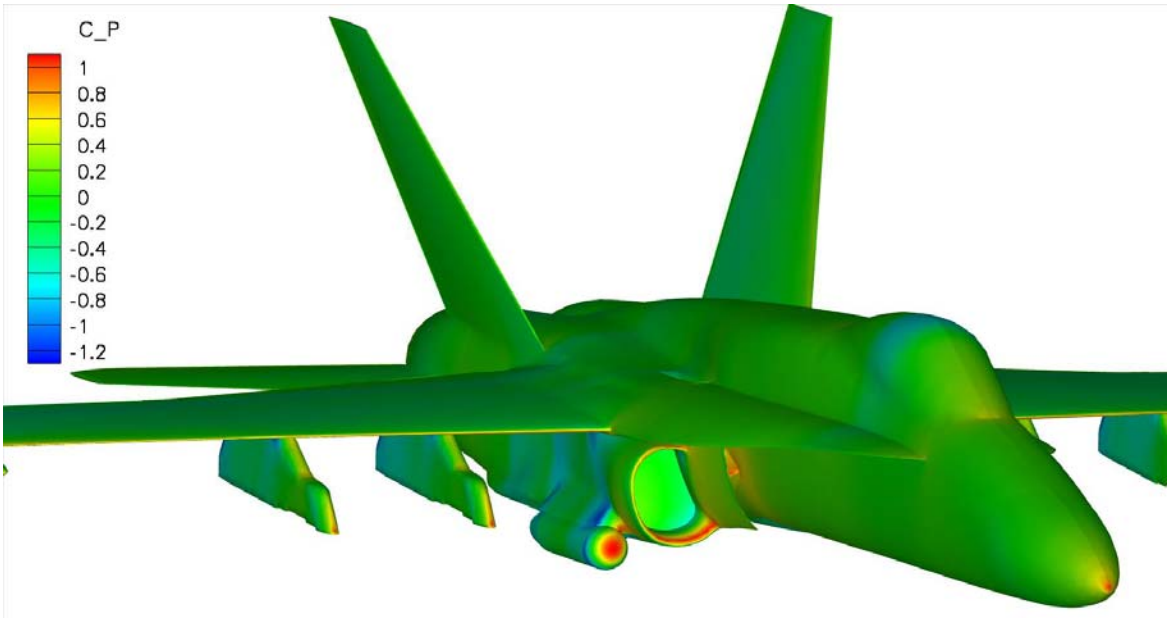


Figure 7. Pressure distribution on F-18C with TFLIR attached

The quantity represented by the colors on the plane's surface is C_P (C_p) instead of actual pressure. C_p represents the pressure coefficient, and is the standard way to non-dimentionalize pressure. The pressure coefficient is defined as follows:

$$C_P = \frac{p - p_\infty}{q_\infty}$$

Equation 4

where p is the static pressure at the point of interest, p_∞ is the freestream static pressure, which is a function of altitude, and q_∞ is the freestream dynamic pressure, which is a function of the ambient density and the velocity of the aircraft. This coefficient allows tests made at different altitudes or airspeeds to be compared to one another. As expected a good portion of the aircraft is green, meaning these regions have a pressure coefficient near zero and a static pressure equal to ambient pressure. Areas where the flow essentially stops or stagnates, such as the nose of the aircraft and the front of the TFLIR, have a high pressure coefficient and therefore a high static pressure. Areas of high curvature have static pressures below that of the ambient flow and thus have negative Cp values. The static pressure distribution along a surface of a shape can be integrated to find the resulting force and moment on the shape due to the flow.

2 COMPUTATIONAL TOOLS

This research project has employed the NASA-developed Tetrahedral Unstructured Software System (TetrUSS) for CFD analysis. This software package is a suite of three different programs, each of which is used in a different step of the process. The CFD research process always starts with manipulation of the geometry, and this part usually begins with Computer Aided Design (CAD) files. These files are three-dimensional computerized models of the geometry to be studied, and can be in many different formats. They are usually created in the design process of the vehicle itself, and as such are not intended for CFD use. It is the job of the first program in the TetrUSS suite, GridTool, to convert CAD files to a suitable format for CFD work.

2.1 GridTool

GridTool is a versatile geometry manipulation program whose primary purpose is the preparation of geometry files for surface grid generation. The majority of the work done at this stage involves the creation of surface patches. A surface patch is a bounded surface consisting of any number of sides used to define the outer boundary of the geometry. Figure 8 below shows the surface patches on the model of the F-18C aircraft used in this project. In the figure, the patch on the front portion of the canopy is highlighted, as indicated by the thicker blue lines with arrows.

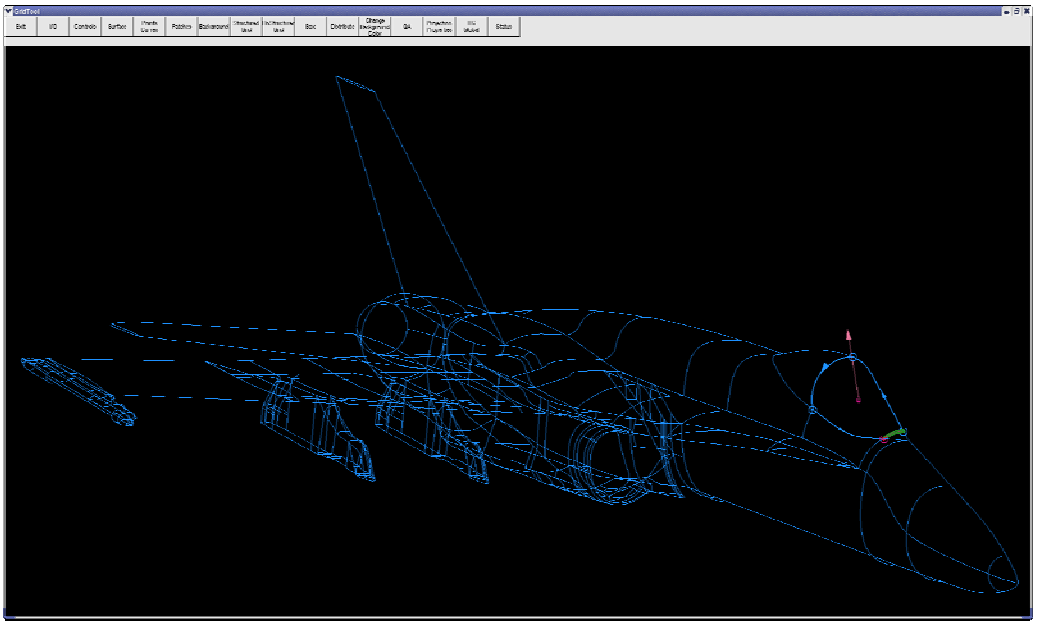


Figure 8. Surface patches used in model of F-18C

Surface patches tell the next program where to create the surface grid, and therefore have defined directions which must point outward from the vehicle. The pink arrow pointing upwards in the center of the selected patch indicates the direction it faces. Establishing a direction is necessary so the grid generation program knows which side of the patch to create the grid on. These patches must also be "watertight," meaning there cannot be gaps between the edges of two adjoining patches. Note from the figure that only half of the aircraft is modeled. This was done to keep the size of the grid smaller and is acceptable because the flow on one side of the aircraft's fuselage has a negligible impact on the opposing side as long as the aircraft is not experiencing sideslip. This makes calculations on the left side of aircraft unnecessary. TetrUSS has the capability to add a reflection plane to allow the use of half models such as this.

Another main function of GridTool is the creation of a background source grid. The size of the cells created in the grid generation step is established by a user-specified source grid. This grid is created in the same coordinate space as the surface patches and is made up of point and line sources, which can have different sizes and magnitude of influence. The background grid used for the F-18C can be seen below in Figure 9. The blue lines represent the surface patches as shown above, while the yellow lines indicate the locations of sources.

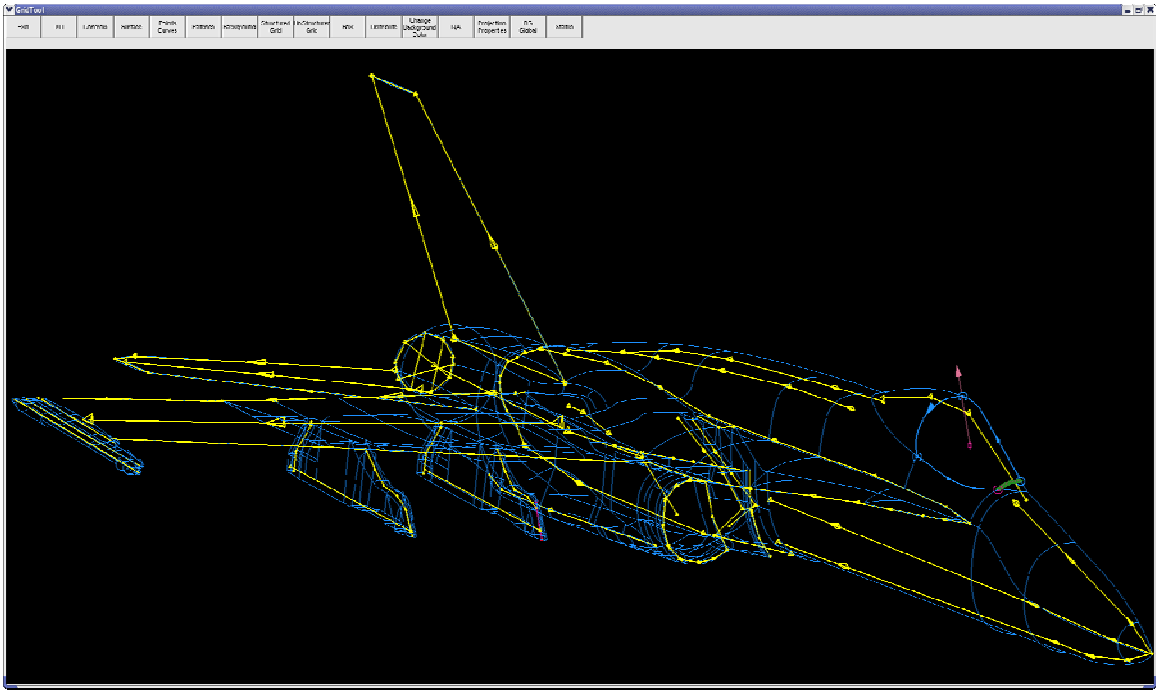


Figure 9. Background sources used in model of F-18C

Proper placement of sources is imperative as they determine the size of the cells which will be generated. It is important to use a sufficient number of cells to properly resolve the flow in areas where flow properties change quickly, such as the leading edge of the wing, but it is inadvisable to use too many cells in less important places as total computational time increases with an increase in number of cells.

For this particular project, a model of the F-18C was available that included the majority of surface patches and background sources. Geometry files were obtained for the TFLIR, ATFLIR, and GBU-31 JDAM (see section 3.3 below), for which surface patches and background sources had to be generated. Once patches were generated on the pods, they had to be attached to the model of the F-18C, which was not a trivial undertaking. Table 1 below shows the number of surface patches used on each model.

Table 1. Number of surface patches used on models

Model	Number of surface patches
F-18C without pod	403
F-18C with TFLIR	417
F-18C with ATFLIR	477
GBU-31 JDAM	79

2.2 VGRID

The second piece of the TetrUSS package is a program called VGRID, which creates the volume grid required for CFD analysis. It uses the output files from GridTool, and begins by creating a background source grid. Using the point and line sources created in GridTool, VGRID first solves a Poisson equation to establish the grid size at every location. This process can be equated to solving a heat transfer problem, in which the point and line sources represent heat sources at different temperatures, and the solution represents a temperature distribution at all locations.[9]

This program is next used to generate a surface grid. Two-dimensional triangular cells are generated sequentially on each surface patch. These triangles are ideally equilateral, but because the TetrUSS package uses an unstructured grid some non-uniformity is permissible. Figure 10 below shows a portion of the completed surface grid of this F-18C.

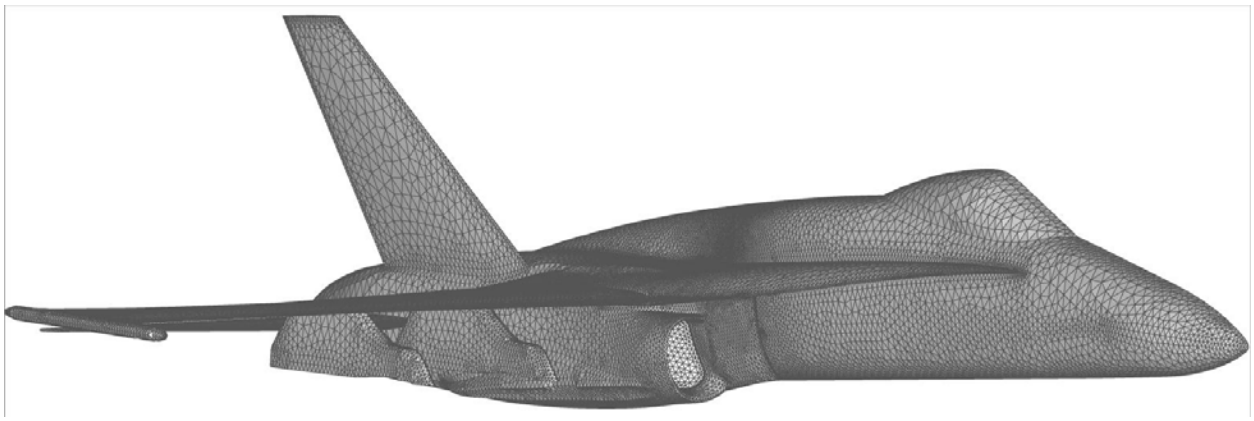


Figure 10. Surface grid of F-18C

[9] Frink, N., Pirzadeh, S., and Parikh, P., "The NASA Tetrahedral Unstructured Software System (TetrUSS)," ICAS Paper 0241, August 2000.

Now instead of the large surface patches created in GridTool, the aircraft is covered with smaller surface patches. These can be better seen in Figure 11 below, which is a magnification of the area around the engine inlet where the TFLIR or ATFLIR would be attached. What is important to note in this graphic is the variation in size of the surface cells. A good grid will use small cells at areas of high curvature and larger cells in flatter regions, where the variation in flow properties should not be as high. In this figure, small cells are used on the leading edge of the pylon and on the inlet cylinder, while larger cells are used on the inlet face and the side of the fuselage.

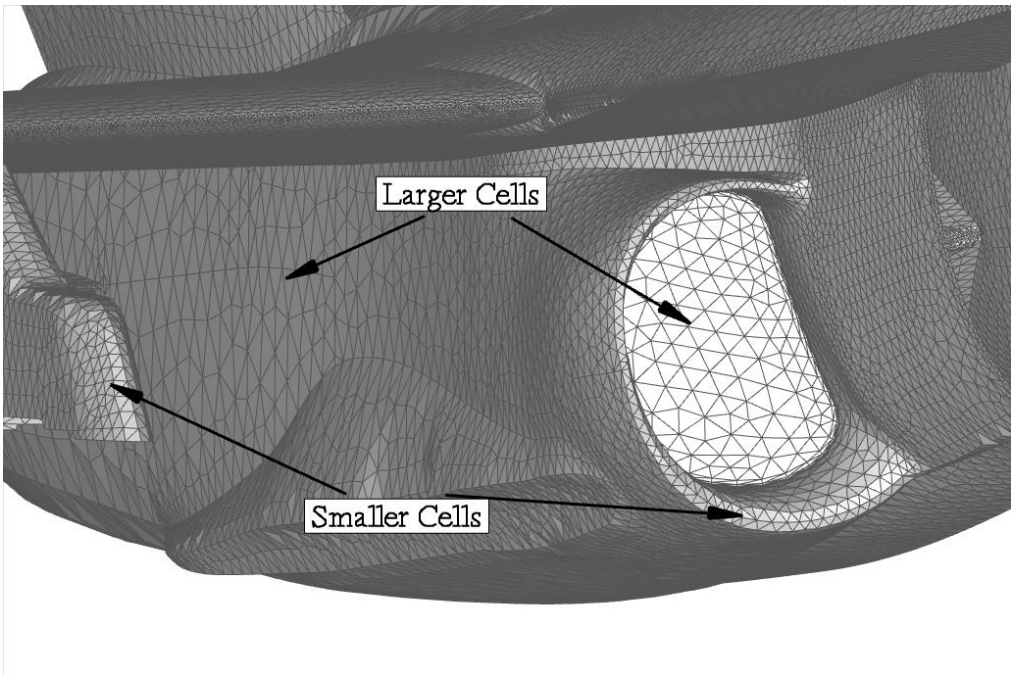


Figure 11. Zoom of F-18C surface grid

The number of cells generated in the surface grids for each configuration tested can be seen below in Table 2. With some variation, as the complexity of the model increased so did the total number of cells.

Table 2. Number of cells in surface grids of analyzed configurations

	Number of surface cells	
Model	**No store attached**	**GBU-31 JDAM attached**
F-18C without pod	58348	77934
F-18C with TFLIR	46082	81174
F-18C with ATFLIR	61707	79852
GBU-31 JDAM freestream	17148	N/A

Once the surface grid is complete, the program will next generate a volume grid. There are two methods in which VGRID can accomplish this. When a viscous result is desired, the advancing layers method is the first to be used. In a viscous flow, the cell properties in the boundary layer close to the surface of the body change very rapidly, and as a result very thin cells are required in order to capture these rapid changes. In order to reduce the overall number of cells required, it is advantageous to keep these cells relatively large in all directions except that normal to the surface. In order to best produce this type of grid, the advancing layers method was developed. This method allows for the creation of relatively ordered cells while maintaining the flexibility of an unstructured grid, and has proven very successful in use. It was first developed and presented for CFD application by Shahyar Pirzadeh in 1993, and its method of propagation can be seen below in Figure 12.

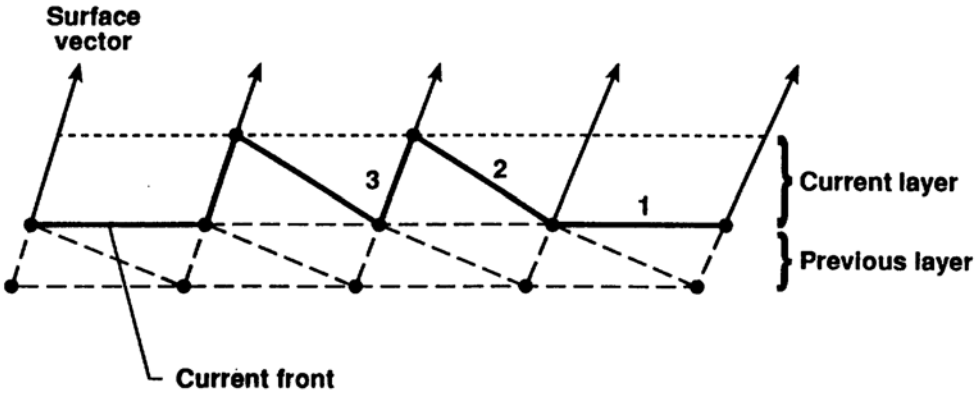


Figure 12. Advancing layers propagation method[10]

Beginning with the surface grid, a surface vector is created from each vertex pointing normal to the surface. The first layer of cells is generated by traveling a specified distance along each surface vector and essentially connecting the dots in the manner shown in the above figure. The first layer of volume cells is generated on each surface cell before the second layer begins, and the second layer is formed in the same manner as the first. The user can specify the initial height of the first cell as well as the rate of growth for the remainder of the layers, which allows great freedom in the number of cells created. It is important to note that this method operates independently of the background source grid; it propagates using only the spacing parameters specified. An example of the viscous layers generated on the F-18C can be seen below in Figure 13. The first layers are far too small to be noticed, but by the end of this method's propagation the layers are quite thick, as can easily be seen on the upper and lower surfaces of the wing.

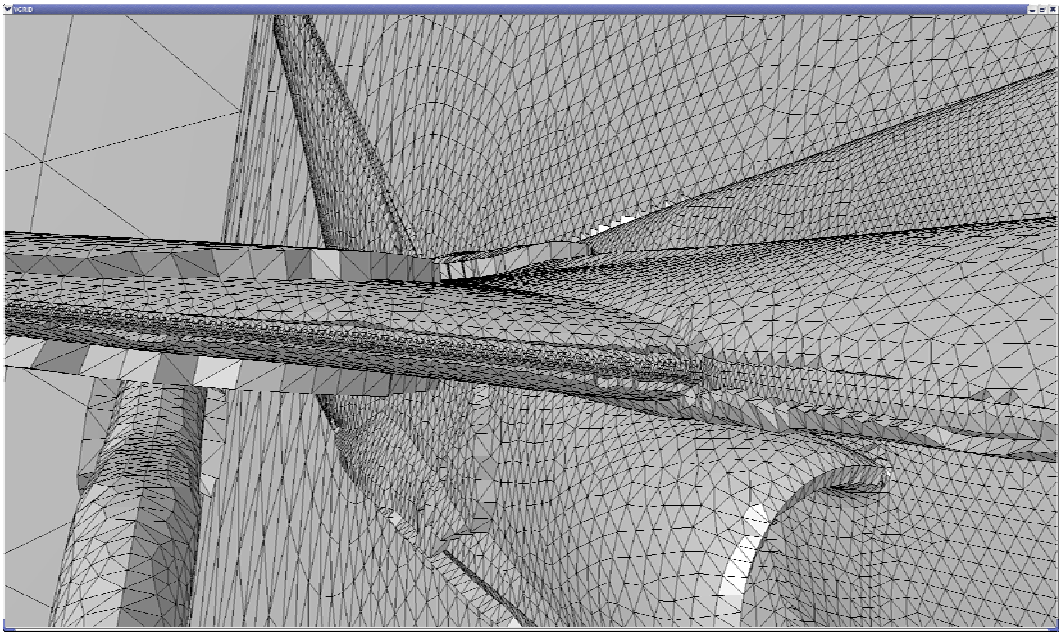


Figure 13. Viscous cells generated with the advancing layers method on the F-18C

The standard method of volume grid generation in VGRID is known as the advancing front. This method is used for generating the inviscid portion of the grid, which comprises the large majority of the volume in any grid. In grids where viscous layers were generated with the advancing layers method, VGRID switches to the advancing front method when the volume of

---

[10] Pirzadeh, S., "Unstructured Viscous Grid Generation by Advancing-Front Method," NASA

the next viscous cell would be greater than the background sources dictate the size should be. The advancing front method attempts to generate equilateral tetrahedra and therefore is not nearly as orderly as the advancing layers method, but has the advantage of greater flexibility which is necessary with complex geometries. Figure 14 shows a two-dimensional grid created with both methods, illustrating the point of transition.

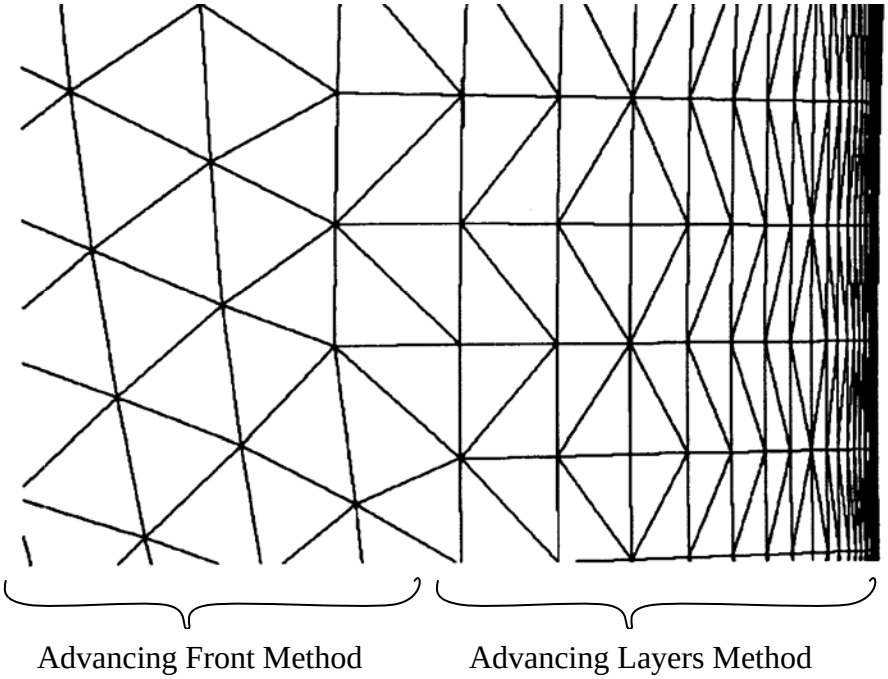


Figure 14. Two-dimensional grid using advancing front and advancing layers methods[11]

The advancing front method also propagates in a different manner. Instead of working layer by layer, it generates new cells along the current front. The current front is defined as the boundary between the gridded and ungridded regions, and is constantly changing as new cells are created. Figure 15 below shows the generation of the volume grid by this method. The majority of grid growth begins at the joint of wing and continues in a somewhat spherical manner until the entire box is filled with cells.

Contractor Report 191449, 1993.

[11] Pirzadeh, S., "Unstructured Viscous Grid Generation by Advancing-Front Method," NASA Contractor Report 191449, 1993.

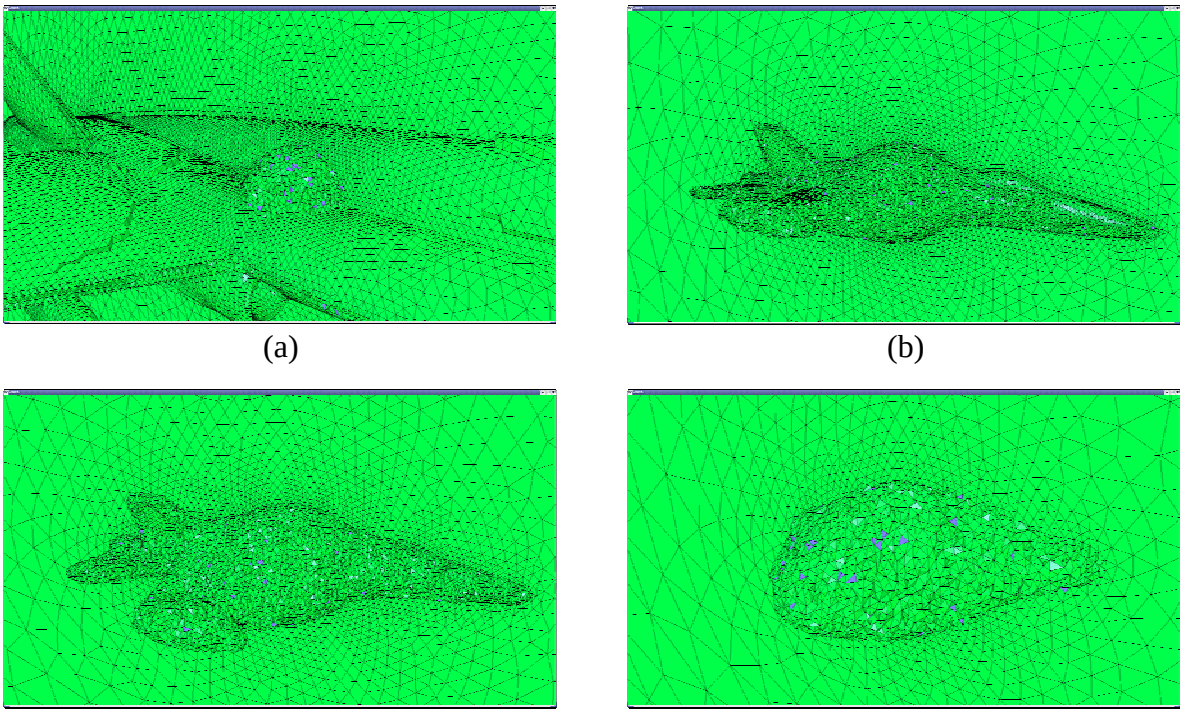


Figure 15. Generation of volume grid by advancing front method

Volume grids were generated in this manner for each configuration in this project, and the total number of cells produced in each can be seen below in Table 3. As expected, the addition of the store to the clean aircraft increased the total number of cells by over 500,000 due to the small size of the cells surrounding the store.

Table 3. Number of cells in volume grids of analyzed configurations

	Number of cells in volume grid (millions)	
Model	**No store attached**	**GBU-31 JDAM attached**
F-18C without pod	1.81	2.37
F-18C with TFLIR	1.63	2.44
F-18C with ATFLIR	1.91	2.47
GBU-31 JDAM freestream	0.92	N/A

2.3 USM3D

The flow solver used in the TetrUSS package is known as USM3D, which was designed for use with unstructured grids. USM3D is a very flexible flow solver, featuring options for different types of flow, inclusion of jet engines or propellers, various face boundary conditions, and the ability to integrate flow properties on a specified set of surfaces, among many others. This last feature is especially important to this research as determination of the forces and moments on the store is the entire goal of CFD calculations. USM3D is a cell-centered solver, which requires fewer tetrahedra and therefore less computer memory relative to the node-centered method.[12] USM3D is also capable of multiple different types of flow simulations. It can perform pure inviscid calculations using Euler's formulation of the flow equations, pure laminar flow calculations, full viscous calculations using the Spalart-Allmaras (S-A) turbulence model, or viscous calculations using a wall function based on the S-A model.

This research has employed the wall function method for multiple reasons. This method operates by applying an analytic function to the sublayer portion of a turbulent boundary layer. This region would otherwise require many very thin cells in order to achieve accuracy, and the removal of these cells has the double benefit of greatly reducing the total number of cells in the grid and removing solution stiffness which is inherent in thin cells. See Frink[12] for more details on the operation of this method.

It is also important to note that flow solutions are not generated in a single step, they require numerous iterations to reach a steady solution, at which point the solution is said to be converged. This type of solution is known as time-marching, and is the result of the chosen numerical method to solve the Navier-Stokes equations. In CFD research convergence is typically evaluated using the concept of residuals, which measure the amount of variation in flow properties from one time step to the next. Experience has shown that solutions can be considered converged when this variation between times is 3-4 orders of magnitude smaller than the original change. This has equated to approximately 1000 to 1500 iterations to reach convergence. Due to the lengthy amounts of time required for a full solution, it is generally wise to examine the value of these residuals over time to ensure that they are decreasing and the solution is

converging. Standard practice is to allow the flow solver to run until the residuals have decreased by at least three orders of magnitude; in this research a reduction in residuals of three orders of magnitude was the primary criterion for convergence. USM3D outputs a file at each time step which contains the value of these residuals, but it is in plain-text format and difficult to analyze. To facilitate this analysis, a program was written in MATLAB which reads the USM3D output file and graphically plots the logarithm of the decrease in residuals versus time. A screenshot of the interface for this program can be seen below in Figure 16, while the MATLAB script for this program can be found in Appendix B.

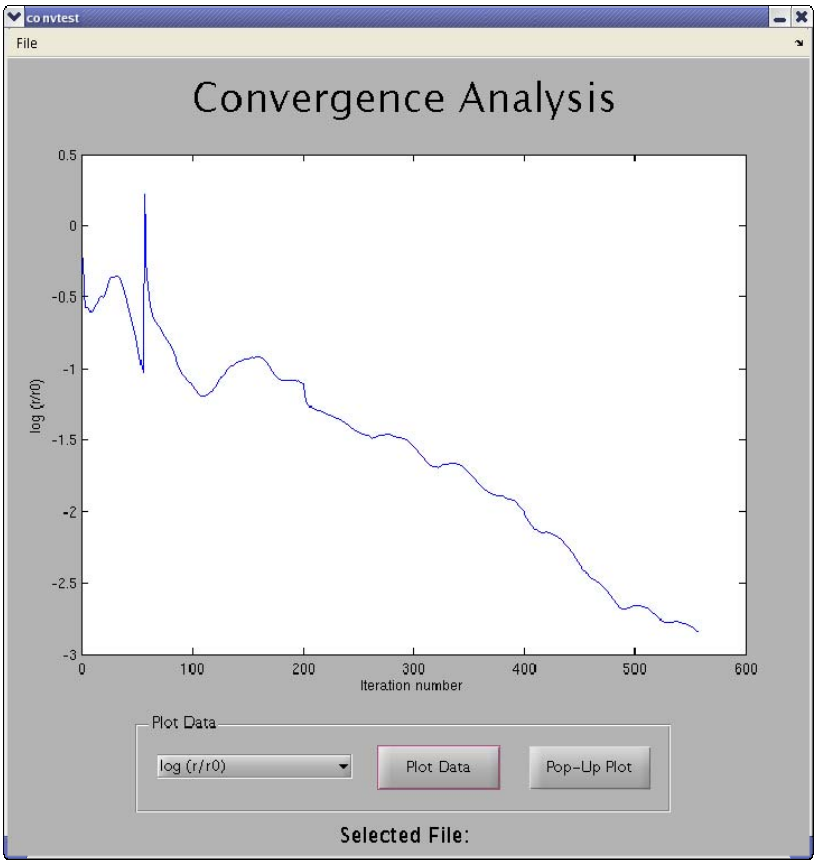


Figure 16. Screenshot of Convergence Analysis program

[12] Frink, N., "Tetrahedral Unstructured Navier-Stokes Method for Turbulent Flows," AIAA Journal, Vol. 36, No. 11, November 1998, pp. 1975-1982.

2.4 Computational Approach

USM3D is a very processor-intensive program and thus required a high end computer to run properly. For the first half of this research the computer used was a Dell Precision 690 featuring two dual-core processors with a speed of 3.0 MHz each and 8 GB of RAM. Using this system, the convergence time for a typical run of USM3D was 27 hours. This time requirement was acceptable at first due to the relatively small number of cases to be solved but proved unacceptably slow when greater numbers of solutions were needed. In order to generate solutions more quickly, the parallel version of USM3D was acquired and installed on a Linux cluster. The parallel version of the code uses multiple processors and allows the total volume grid to be broken into a number of smaller sections, each of which is solved by a single node. Using eight nodes on the Linux cluster, the time required per solution fell from 27 hours to 5 hours, allowing many more solutions to be generated in a given amount of time.

The switch from series to parallel processing was also accompanied by an upgrade to the USM3D software. The version first used in series was v5.3, while the parallel version was v6.0. The upgraded versioned features better handling of flow separation, among other improvements.

3 COMPUTATIONAL ANALYSIS

A systematic approach has been used throughout this research project in order to produce accurate and replicable data. The first phase consisted of validation of the F-18C model using previously generated wind tunnel data. The validation metric was a comparison of the flow properties in the vicinity of the store carriage location. The TFLIR and ATFLIR were then added to this model and their effects on the flowfield were analyzed. The model of the GBU-31 JDAM was obtained and validated next, again using existing wind tunnel data. The validation metric was the normal force and pitching moment on the store. After validation, the JDAM was attached to the F-18C both with and without the targeting pods attached, and the various solution results were analyzed.

3.1 F-18C Model Validation

The first step in the research process consisted of manipulating the geometry of the aircraft to be used, the F-18C Hornet. This aircraft was chosen because it was involved in the initial mishap and is one of the principle platforms that uses the TFLIR and ATFLIR pods. Fortunately, the geometry of this aircraft had already been used with GridTool, so very little work was needed to prepare the model for grid generation. Using VGRID and USM3D, solutions were soon generated for the clean (no pods or stores) configuration at Mach numbers of 0.8 and 0.95. These speeds were chosen to allow comparison with wind tunnel data previously obtained at these Mach numbers. The particular wind tunnel data used were generated in tests at the David Taylor Research Center transonic wind tunnel in 1989-1990. This tunnel is one of the few large-scale transonic wind tunnels in this country, featuring a test section that is 7 feet high and 10 feet wide.[13]

The measurements taken in this study were recordings of the upwash and sidewash angles at various points in the flow. The upwash angle (α) is defined as the angle between the

[13] Cenko, A., "Configuration Effects on the F-18 Aircraft Flowfield." NAVAIR Report No. NADC-90111-60, May 7, 1990.

vertical and axial components of the airflow, and is measured with a slender probe positioned at a desired location in the tunnel. The sidewash angle (β) is similar but uses the normal component of the velocity, which points out the right wing, instead of the vertical. Because the axial component is much larger than both the normal and vertical components, these angles will rarely exceed 5º in magnitude. In the wind tunnel experiment these angles were measured along an axial flow line beneath the inboard pylon. Figure 17 below shows where this line was located in reference to the aircraft, with the flow line colored red. This same flow line was analyzed in this project using CFD in order to facilitate comparison.

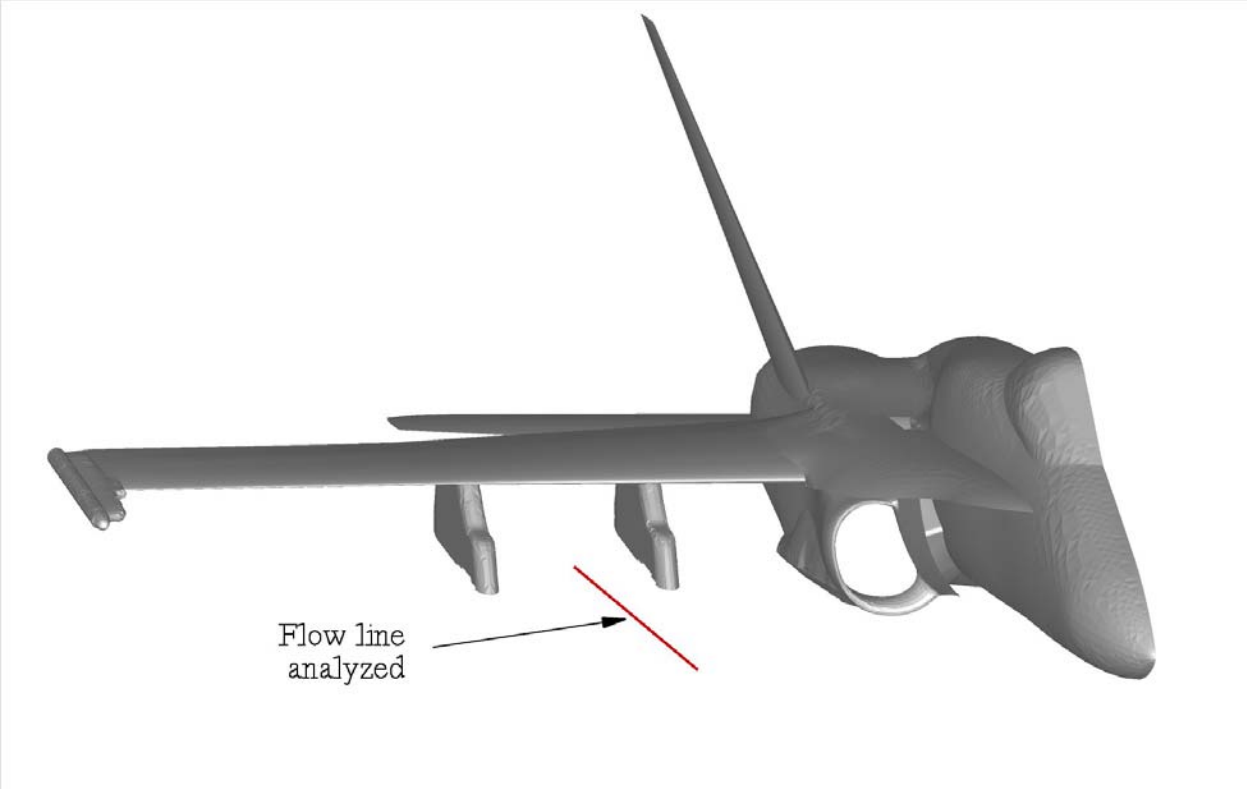
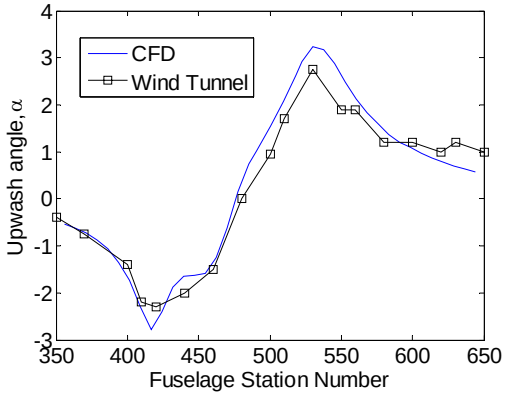
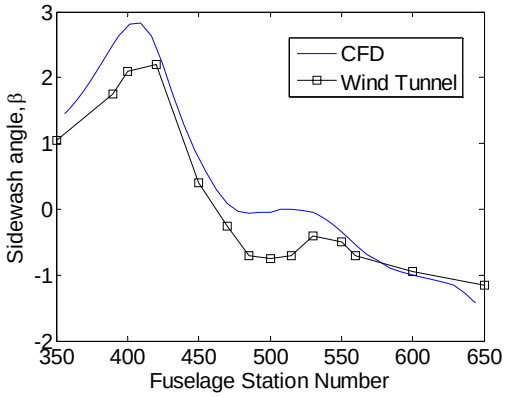


Figure 17. Axial flow line beneath inboard pylon

Analysis of the CFD solutions was accomplished using the post-processing software Tecplot®, which enabled extraction of the flow velocity components along the same flow line tested in the wind tunnel. The CFD results were then plotted against the wind tunnel results, and the resulting plots can be seen below in Figure 18 and Figure 19, which are at Mach 0.8 and 0.95 respectively.

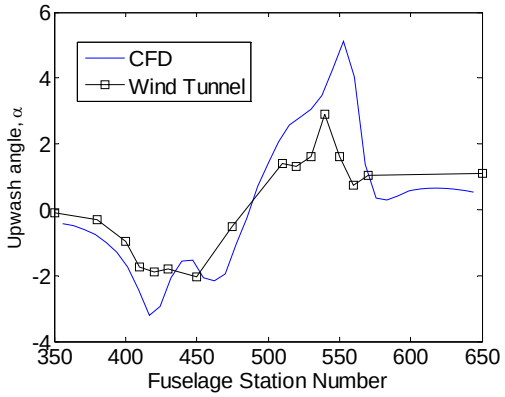


(a) Upwash angle at Mach 0.8

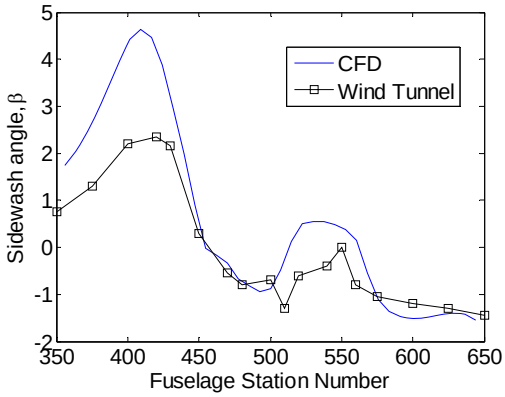


(b) Sidewash angle at Mach 0.8

Figure 18. Comparison of CFD and wind tunnel upwash and sidewash angles at Mach 0.8



(a) Upwash angle at Mach 0.8



(b) Sidewash angle at Mach 0.8

Figure 19. Comparison of CFD and wind tunnel upwash and sidewash angles at Mach 0.95

The horizontal axis of these figures is the fuselage station number, which represents the axial flow line shown in Figure 17 above and is measured in inches aft from the nose of the aircraft. Station 350 corresponds to the leading edge of the wing, while 650 represents the trailing edge. As can be seen, both upwash and sidewash angles calculated with CFD correlated well with the wind tunnel data at Mach 0.8, while the results at Mach 0.95 did not match as well. The most important discrepancy is the initial peak in the sidewash angle around station number 400, in part (b) of Figure 19. Recall that the initial incident occurred because the bomb yawed outward upon release and the tail fins impacted the aircraft. Sidewash angle is the best way to estimate the degree of yaw that a store will experience short of actually testing the store. It was

therefore very important to match the sidewash angle of the clean F-18C before proceeding further.

The model at this point had a “flow-through inlet,” meaning that an open tube in the model connected the intake to the exhaust of the engine. This setup was sufficient at lower Mach numbers, but began to cause problems in the transonic region. This occurred because the area of the tube and the exhaust plane were both smaller than that of the intake. When flow is subsonic, the velocity of a flow increases with decreasing area in order to maintain a constant mass flow rate. This is why nozzles tend to have smaller areas than the hoses they are attached to. This only works below the speed of sound, though. When flow reaches Mach 1.0 it can no longer accelerate. When the flow in the flowing inlet reaches Mach 1.0 at some point, it is considered choked, at which point the mass flow rate can no longer increase. At Mach 0.95, the flow through the engine did choke, which resulted in less air moving into the inlet than would actually occur. The excess air had to move somewhere, and some of it moved outward, thus inflating the sidewash angle beneath the pylon.

The solution to this problem required a model of the engine. The TetrUSS software package allows for the modeling of jet engines and includes a parameter, known as jet total pressure, which affects the mass flow across the engine inlet plane. Thus, adding this engine model created the ability to specify the amount of mass flowing into the engine. Figure 20 below shows the effect of increasing the mass flow across the inlet of the engine.

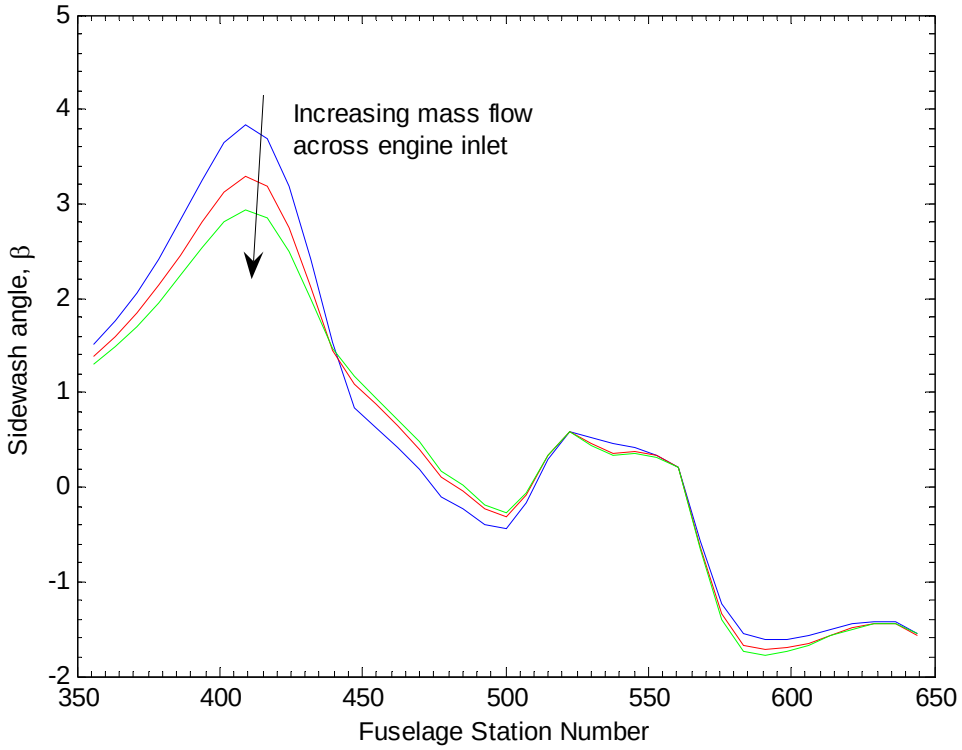


Figure 20. Effect of varying engine mass flow on sidewash angle at Mach 0.95

As this figure shows, increasing the mass flow across the engine does decrease the magnitude of the sidewash angle at station 400. Additionally, a point was reached at which further increases in jet total pressure no longer had an effect. At this condition the inlet was ingesting all of the mass impacting the inlet face, and this value was chosen for future use. Figure 21 below shows a comparison of the sidewash angle at Mach 0.95 between the CFD solution with engine model and the wind tunnel data.

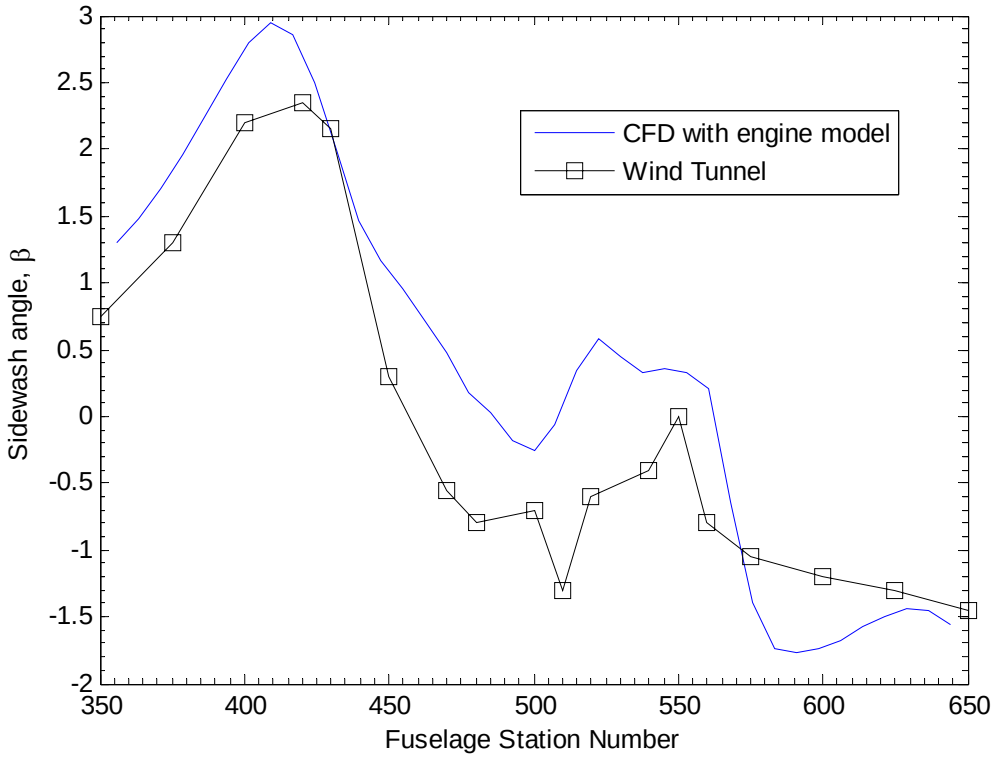


Figure 21. Sidewash angle comparison using engine model and wind tunnel at Mach 0.95

While there are still discrepancies between CFD predictions and wind tunnel results, the difference is now on the order of 0.5 degrees instead of 2.0 degrees. NAVAIR store separation engineers consider a half-degree variation good agreement. At this point the CFD model of the clean F-18C aircraft was considered valid and saved for further use.

3.2 Targeting Pod Flowfield Effects

The next step following validation of the aircraft model was to attach the targeting pods and examine their effect on the same axial flow line used to validate the clean F-18C. The pod geometries were obtained as computer aided design files and had not been prepared for use with CFD. The next task, therefore, was to create surface patches and background sources using these CAD files, followed by the attachment of these pods to the F-18C. These CAD files were not developed with CFD use in mind and significant time was necessary to properly manipulate

these files into the correct format. A portion of each completed surface grid can be seen below in Figure 22 and Figure 23 , with the geometric differences between the two pods emphasized.

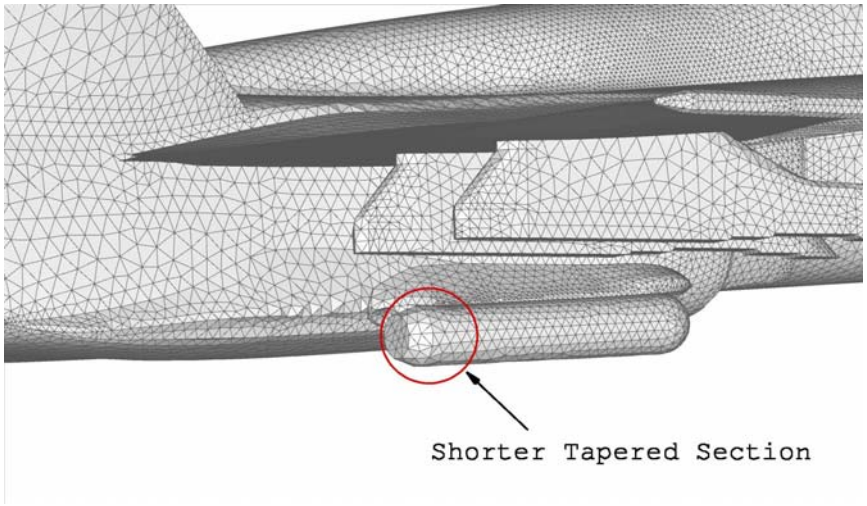


Figure 22. Surface grid of ATFLIR pod attached to F-18C aircraft

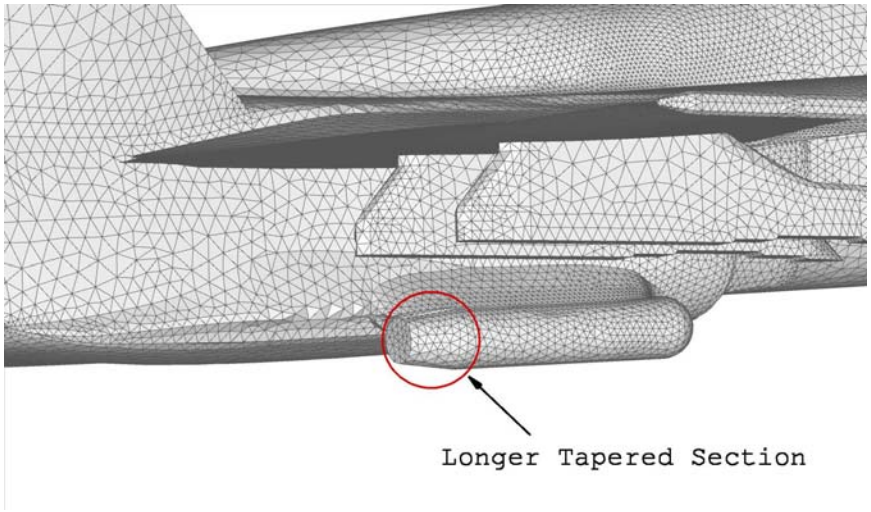


Figure 23. Surface grid of TFLIR pod attached to F-18C aircraft

Initially, it was thought that the important physical difference between the pods was the shape of the leading edge. The leading edge of the TFLIR is blunter and set farther aft, while that of the ATFLIR is more streamlined and extends farther forward. Later testing, however, revealed that the variation in aft tapering was the important distinction between the two pods. As

shown, the ATFLIR has a much shorter tapered section, while the TFLIR's is longer and more gradual. Other than these minor differences, the two pods are nearly identical.

After successful grid generation, the flows around both pods were solved and data were extracted along the same axial flow line as shown in Figure 17. The results in upwash were first examined, and it was found that the addition of pods had virtually no effect on the upwash angles along the flow line. The effect of the pods on sidewash angle, however, was very significant. Figure 24 shows the details of this effect.

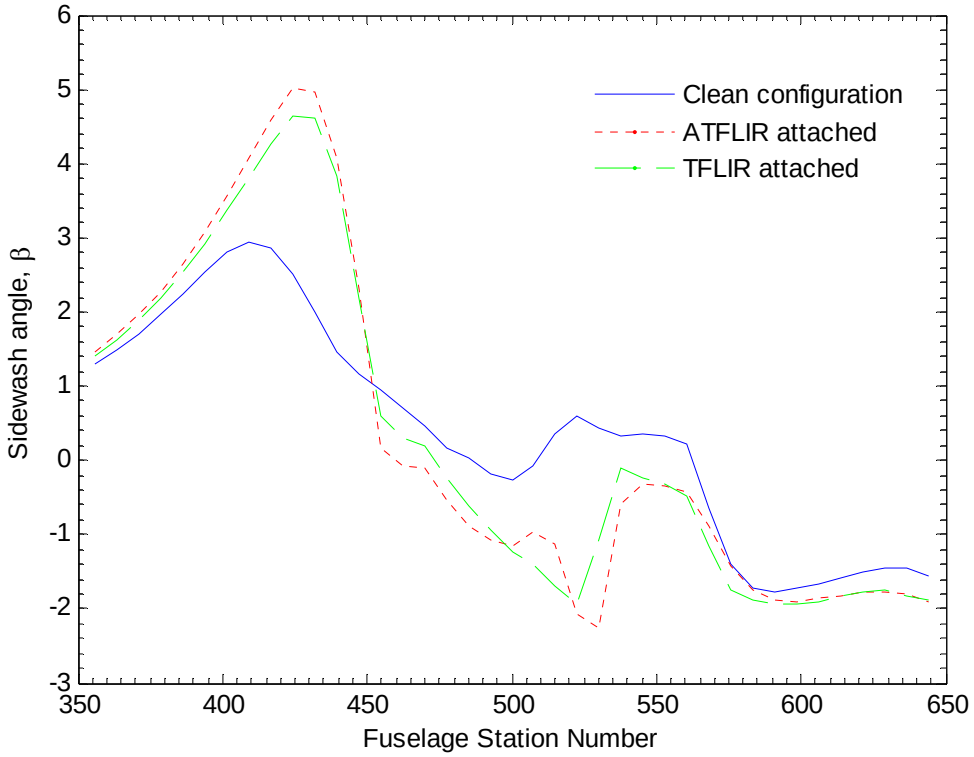


Figure 24. Effect of ATFLIR and TFLIR on sidewash angle along axial flow line at Mach 0.95

As this plot shows, the addition of a targeting pod dramatically increased the peak value of sidewash angle around station 425 and then decreased it near the center of the wing. It is also clear that the two pods had similar effects on the flow but were not identical. Based on slightly higher and lower peak values, this plot indicates that the ATFLIR would have a slightly greater effect on the yawing moment of a bomb beneath this pylon. While they were by no means

conclusive, these results were encouraging in that they provided a qualitative indication that the pod's effects on flow sidewash angle were significant.

This portion of the analysis required the generation of many plots such as those seen in Figures 18-21. In order to decrease the repetition involved in generating these figures, a MATLAB program was written to speed the process. This program is similar to the one used to analyze convergence, but allows for the input of multiple data files to be plotted simultaneously and has wind tunnel data built in. A screenshot of the interface for this program is shown in Figure 25 and the associated MATLAB script can be found in Appendix C.

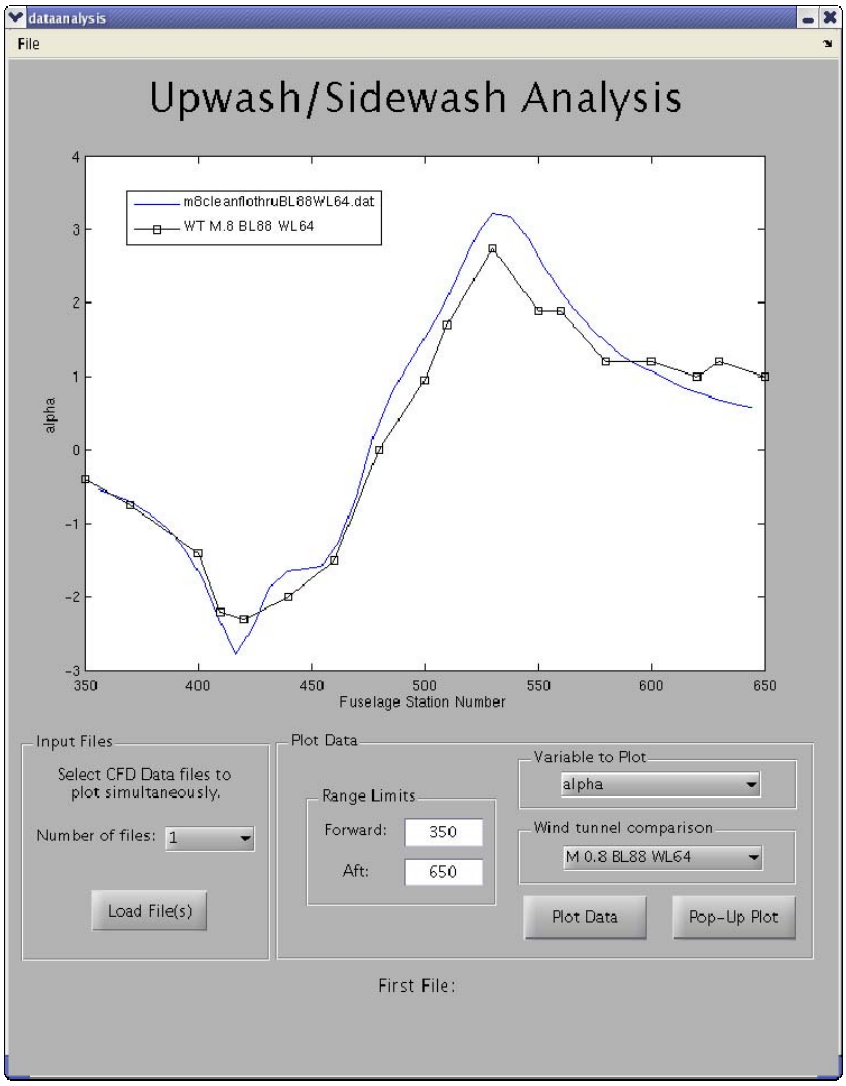


Figure 25. Upwash/Sidewash analysis program interface

3.3 GBU-31 Model Validation

While the axial flow line analysis indicated that the TFLIR and ATFLIR pods had a large effect on the flow, the only way to actually demonstrate this was to introduce a store to the model and calculate the forces and moments generated on it. The store chosen for use in this analysis was the GBU-31 Joint Direct Attack Munition (JDAM). An image of this bomb can be seen below in Figure 26.

Figure 26. GBU-31 Joint Direct Attack Munition[14]

The GBU-31 JDAM is an upgrade of the Mark 84 unguided bomb and has a standard weight of 2000 pounds, making it one of the heaviest fighter-carried weapons. The JDAM upgrade added a guidance package including an inertial navigation system, GPS receiver, movable tail control fins, and the strakes along the sides. The parts in white in Figure 26 are all components of the JDAM package. This bomb was one of the principle stores used in the flight test program, so there are telemetry data with which to compare a predicted trajectory. The geometry file obtained for this bomb was in CAD format and thus required extensive manipulation to prepare it for use with CFD. The completed surface grid for this bomb can be seen below in Figure 27.

[14] "Joint Direct Attack Munition," http://en.wikipedia.org/wiki/Image:GBU-31_xxl.jpg#filehistory, (November 25, 2007).

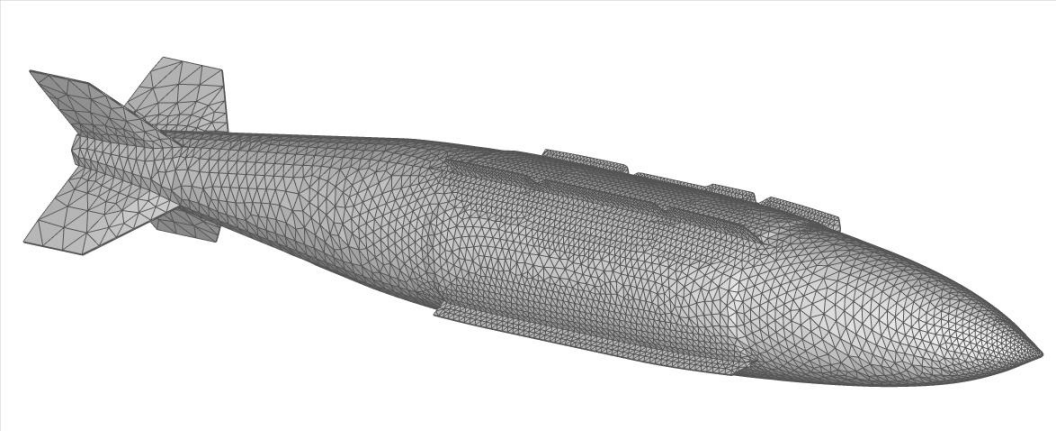


Figure 27. Surface grid of GBU-31 JDAM

Before adding the GBU-31 to the F-18C aircraft, it was necessary to validate the model of the store, again by comparison to wind tunnel results. Freestream data, in which the store is tested without an aircraft, had been previously generated in multiple large-scale wind tunnels. In these tests, the aerodynamic forces and moments generated on the store are measured with a balance. In the field of aerodynamics, it is common practice to then non-dimensionalize the recorded forces and moments so that data can be compared with similar experiments at different speeds or scale. Forces are non-dimensionalized with respect to the dynamic pressure of the flow and a reference area, while moments are taken with respect to dynamic pressure, a reference area, and a reference length. The resulting values are known as force and moment coefficients and are the typical means through which aerodynamics are discussed. Equation 5 below shows the calculation of a typical force coefficient, in this example for normal force.

$$C_N = N \big/ qS$$

Equation 5

Where C_N is the lift coefficient, N is the total lift force, q is the freestream dynamic pressure, and S is a reference area. Equation 6 below represents a moment coefficient and varies only by the parameter c, which is a reference length.

$$C_m = M \big/ qSc$$

Equation 6

Significant amounts of wind tunnel data have been generated at Mach 0.95, which is in the transonic region that is the focus of this project. Validation of the model was therefore undertaken at this speed using the method of an angle-of-attack sweep. In the wind tunnel, force and moment data points had been taken at numerous different angles of attack (AOA), and for the purpose of comparison, multiple angles of attack were computed using CFD as well. The two important parameters recorded were the normal force coefficient, which measures the force generated in the vertical direction, and the pitching moment coefficient, which indicates the magnitude and direction of the bomb's tendency to pitch up or down. It is important to note that the JDAM's symmetry allowed the AOA results compared here to be directly equated to sidewash (β) results. This means, for example, that the pitching moment produced at a given angle of attack is the same as the yawing moment that would be produced by that same angle of sidewash. USM3D determines these parameters by first solving for the flow properties in every cell, and then integrating the pressure distribution on each surface cell to determine the total force and moment about the bomb's center of gravity. Figure 28 shows the normal force coefficient generated at multiple angles of attack.

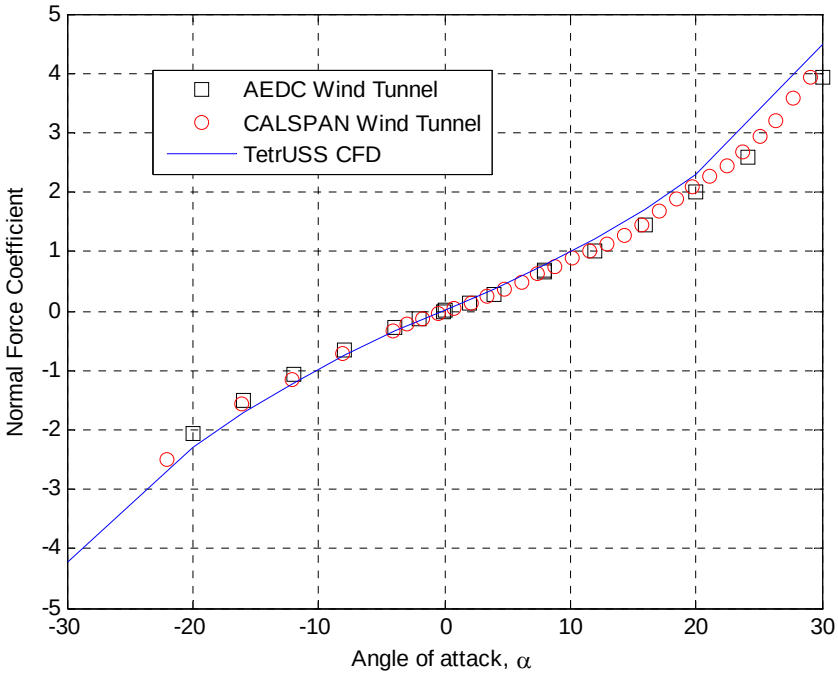


Figure 28. GBU-31 normal force coefficient at multiple angles of attack

This figure indicates good correlation between the CFD results and the measured wind tunnel data. The CFD slightly overpredicted the magnitude on both the positive and negative ends of the plot, but not enough to cause concern. Figure 29 below shows the other parameter: pitching moment coefficient. For the majority of angles of attack CFD shows good correlation, once again very slightly overpredicting the magnitudes. Around positive and negative 15º AOA, though, the wind tunnel data indicates a departure from the linear behavior that is not captured by CFD. This behavior may be caused by vortices generated on the strakes impacting the tail fins. Increased grid resolution may serve to capture this effect; alternatively, this may be caused by interference effects of the apparatus holding the store in the wind tunnel.

Figure 28 and Figure 29 highlight the fact that wind tunnel testing and CFD analysis are not exact duplicates of each other. Each has their own strengths and weaknesses. Furthermore, even the exact geometries of the test articles in each case are slightly different than that of the actual store. Engineering approximations are made in the manufacture of the wind tunnel model as well as in the geometry definition of the CFD model.

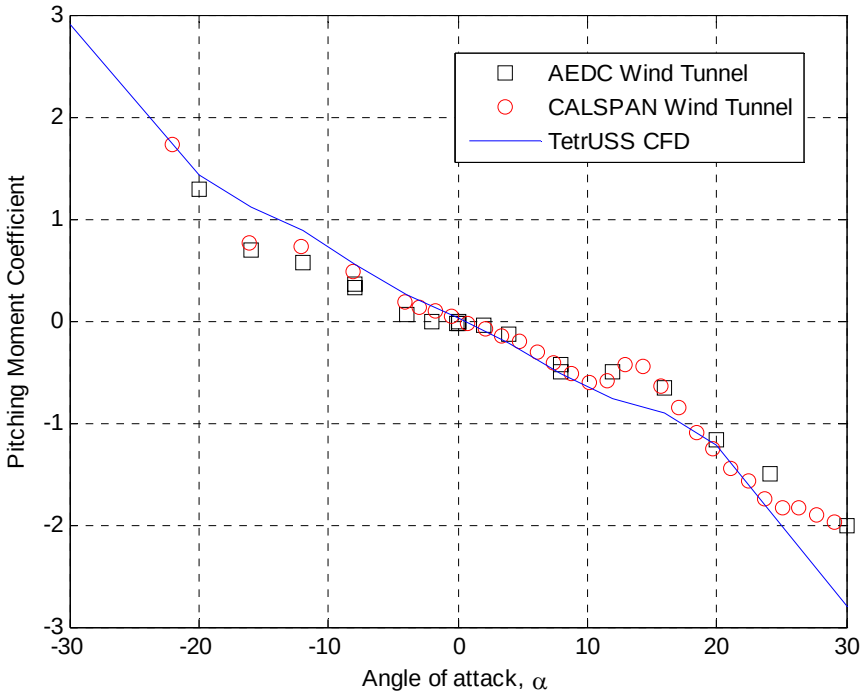


Figure 29. GBU-31 pitching moment coefficient at multiple angles of attack

Based on comparisons with large-scale wind tunnel data, this model of the GBU-31 JDAM has been considered accurate. While the variation at 15º AOA has the potential to adversely affect results it is possible that the CFD result is more accurate than the wind tunnel model; regardless, the remainder of the angle-of-attack range exhibits very close correlation.

3.4 Carriage Position Analysis

Once the models of the store and aircraft had been created and validated, the store was placed in the carriage position beneath the inboard pylon of the F-18C. Carriage position refers to the condition when the bomb is attached to the aircraft, and is specified for each type of store on every aircraft. Figure 30 shows the surface grid of the GBU-31 in its carriage position without a targeting pod attached to the aircraft.

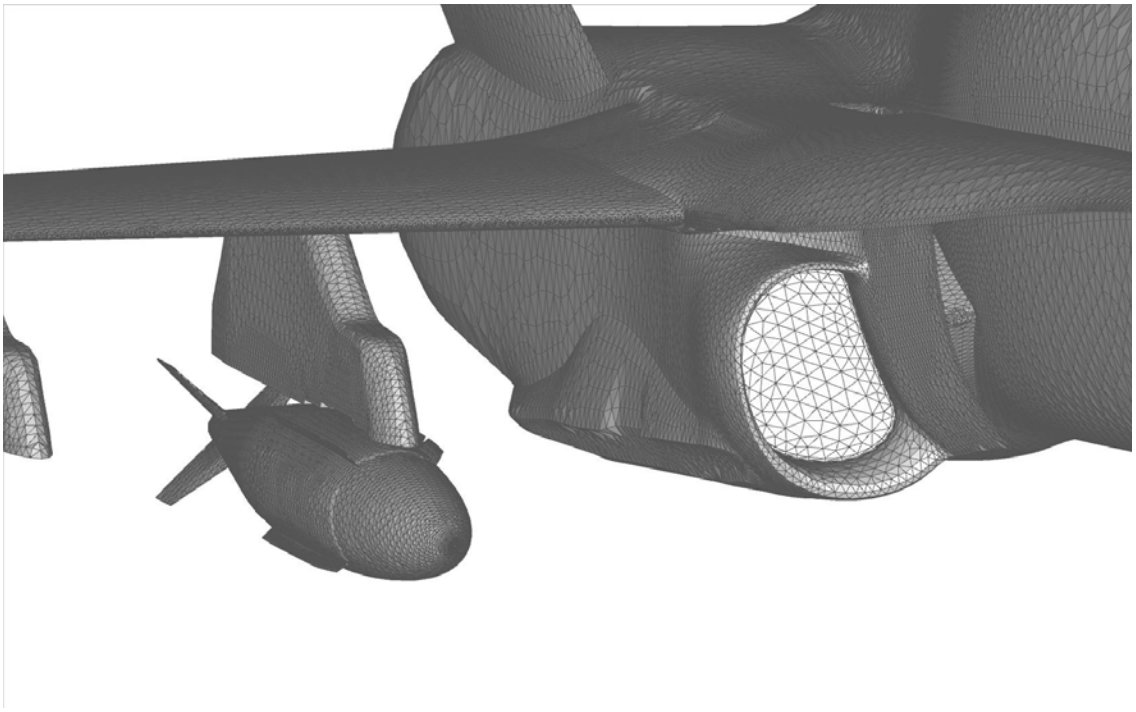


Figure 30. Surface grid of GBU-31 JDAM in carriage position with no pod

At this point in the research it was decided to add Northrop Grumman's Litening pod to this study in addition to the TFLIR and ATFLIR. The Litening pod was designed for the same purpose as the other two pods, though it incorporates some additional features and is therefore

larger than the TFLIR and ATFLIR. A picture of the Litening pod attached to an F/A-18A can be seen in Figure 31, while the surface grid is shown in Figure 32

Figure 31. Litening pod on underbelly of F/A-18A Hornet

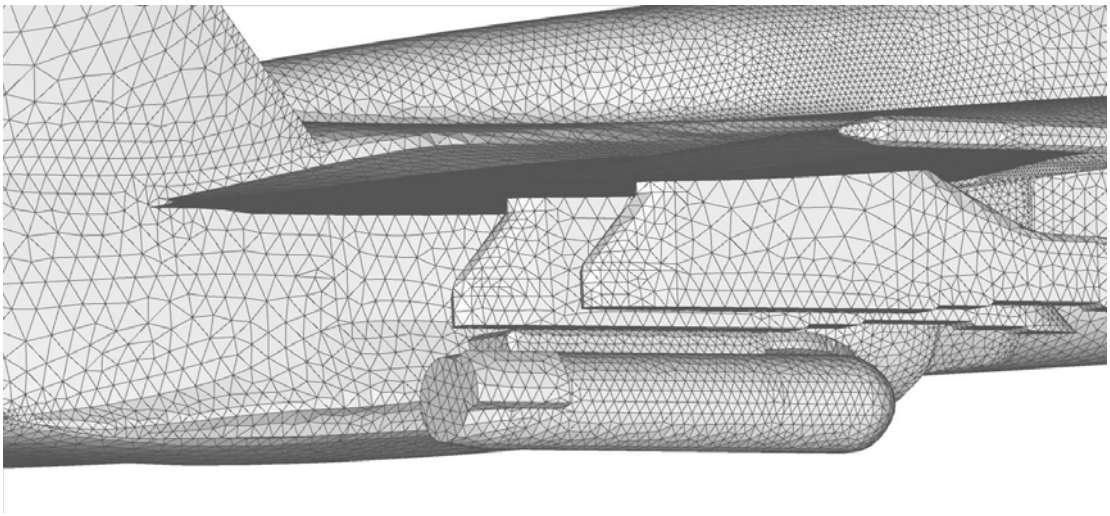


Figure 32. Surface grid of Litening pod attached to F-18C Hornet

Note that the Litening pod is larger than both other pods in length and diameter. The image of the grid shown in Figure 32 was taken from the same angle as those of the TFLIR and ATFLIR pods in Figure 22 and Figure 23, allowing for a comparison in size. It also features a decidedly un-aerodynamic protrusion towards its aft end that was expected to have some influence on store separation, though what the effect would be was not known.

The flight test program described in Rothback had concluded that the worst-case condition was at 0.0° aircraft AOA, so this is the condition that was tested throughout this portion of the research. This same flight test data showed that the initial trajectory of the store is very dependent upon the Mach number at which the store is released, and as a result the forces and moments on the GBU-31 were found at various Mach numbers in the transonic flight regime. Each Mach number at each configuration required a separate run of USM3D, which took approximately 27 hours each using the series version of the code. This was the most computationally intensive portion of this research effort. Thirteen Mach numbers were chosen in the range of 0.8 to 1.05 and all four configurations, ATFLIR, TFLIR, Litening, and clean (no pod), were examined. Resultant force and moment data were extracted for each test and the results are presented in tabular form in Tables 4 through 7 in Appendix A. The two most important factors, pitching moment and yawing moment, are presented graphically in Figures 33 and 34 below. These two factors are most significant because store trajectories resulting in aircraft impact are associated with dramatic changes in the pitch and yaw angle of the store. The pitch and yaw of the store alone may be enough to swing a portion of the store (typically its tail) into the aircraft, or it may result in lateral or vertical motion that will move the store into a collision path with the aircraft.

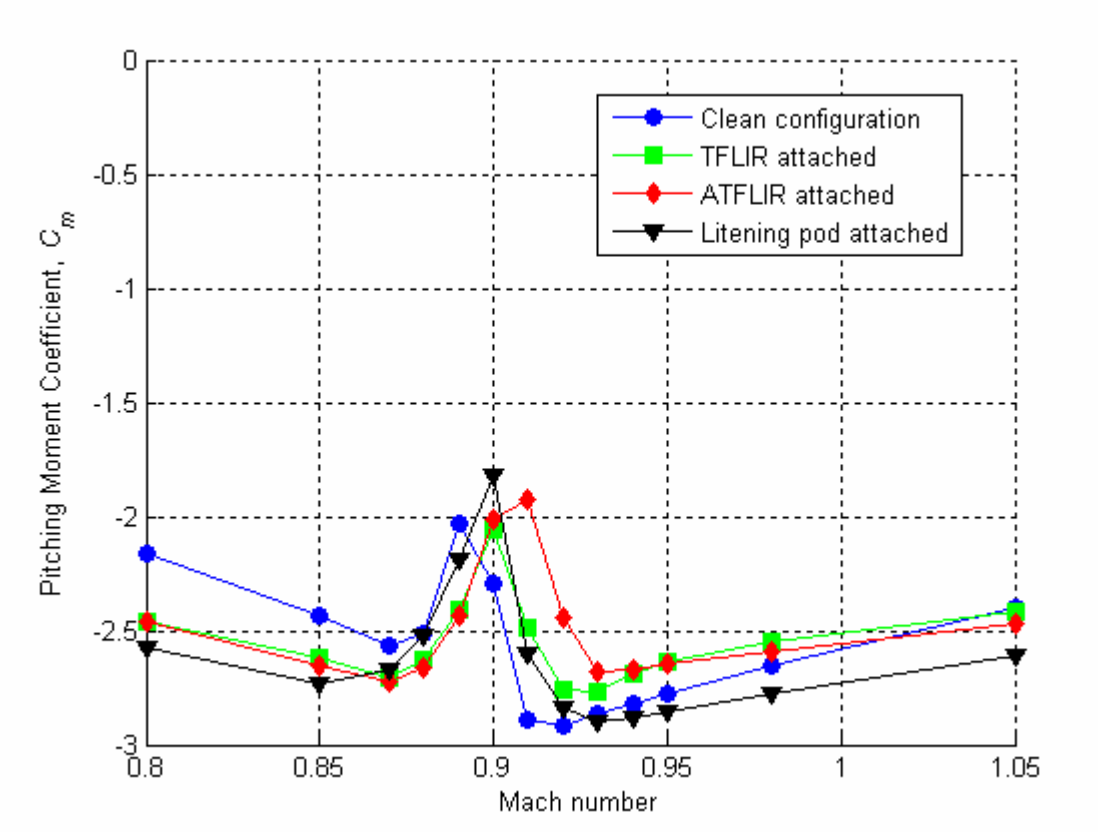


Figure 33. GBU-31 JDAM pitching moment coefficient in carriage position

As Figure 33 shows, the addition of the targeting pods does not have a very significant effect on the pitching moment of the GBU-31 in carriage position. Generally, any nose up motion of the store in the vicinity of the aircraft is dangerous. Some negative store pitching moment, which causes the nose to rotate downward, is favorable for store separation applications because the store will leave the vicinity of the aircraft more quickly. Excessive nose-down store pitching moment may not be desirable as the motion could cause the tail of the store to strike the pylon it was attached to. The store is forced away from the F-18C aircraft using fore and aft pyrotechnic ejector charges. The magnitude of the ejector force can be varied somewhat depending on the choice of pyrotechnic ejector charges and in this way the store separation engineer has some control over the pitching motion of the store. One set of ejector forces, however, must work over the entire range of possible release conditions. The region in which there is variation in these data is between Mach 0.90 and 0.92. The general trend in pitching moment change with Mach number is qualitatively similar for all configurations. The region of

rapid change varies slightly for each configuration with the clean configuration spiking first and the ATFLIR configuration spiking last. In all cases, the change is on the order of 1.0 from minimum negative value to maximum negative value. This change in magnitude is significant, but overshadowed by the behavior of the yawing moment which is explored next.

Figure 34 below plots yawing moment coefficient against Mach number for each of the four configurations. As can be seen, the addition of the targeting pods has a tremendous effect on this parameter.

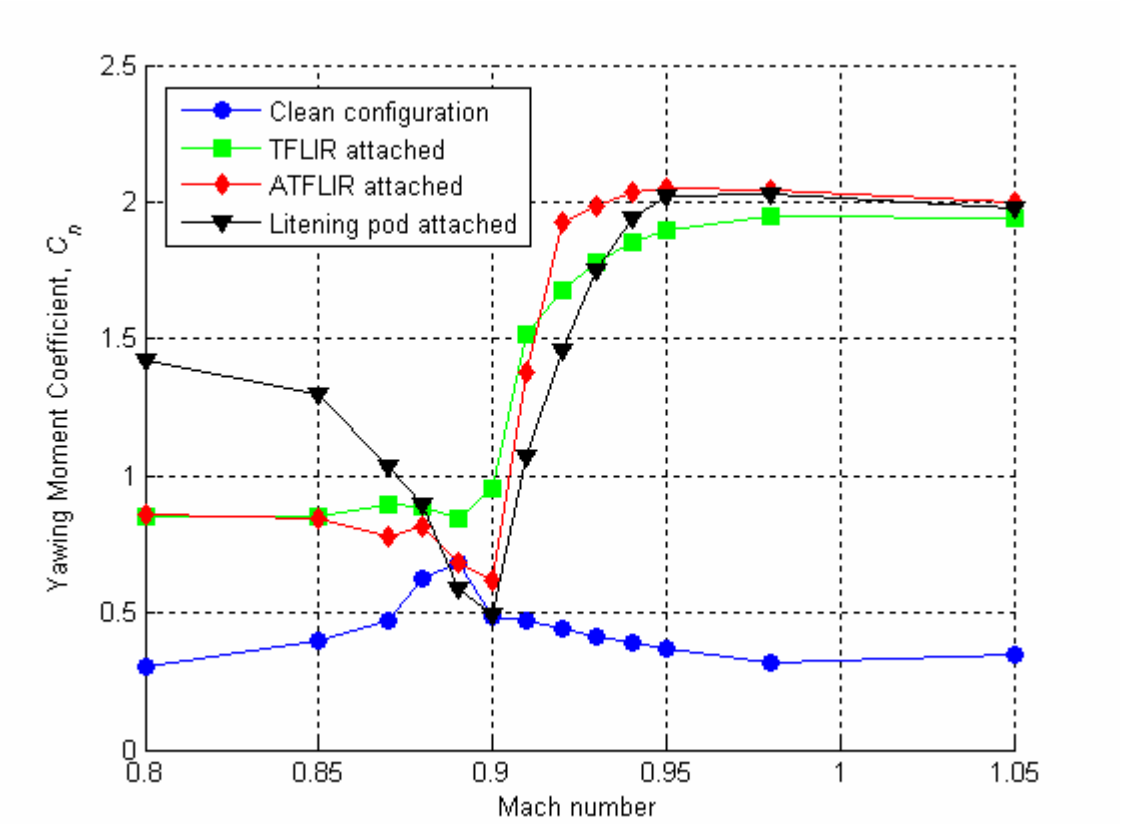


Figure 34. GBU-31 yawing moment coefficient in carriage position

In the clean configuration, the yawing moment coefficient does not vary significantly with Mach number. There is a slight peak at Mach 0.89 but this is not much greater than its value at all other Mach numbers. Clearly, the most significant aspect of this plot is the dramatic increase in yawing moment coefficient that occurs between Mach 0.90 and 0.94 in the conditions with pods attached. Over this period the coefficient rises from around 0.5 to 2.0, an increase of 300%. This increase in positive yawing moment will initially cause the bomb's nose to rotate

outboard and its tail to rotate inboard. Recall that in the initial flight incident, the problem occurred when the bomb's tail fins impacted the TFLIR pod. This was the result of just such a yawing moment as predicted by these data.

3.5 Trajectory prediction

The final steps in this store separation analysis were to predict the store's movement following separation and to compare this prediction to flight test data. Trajectory predictions were made using the Navy Generalized Separation Package (NAVSEP), which is a software program that computes a store's position and orientation following its release. This program incorporates aerodynamic forces and moments and allows complex inputs such as initially restricted motion and ejector force modeling.

A computed trajectory is defined as knowledge of all six degrees of freedom at set time intervals. These include three position components in the axial, horizontal, and vertical directions, which are referenced to an established frame. Three orientation variables are also needed and are defined as ψ (psi), θ (theta), and φ (phi), which respectively are the yaw angle (positive nose right), the pitch angle (positive nose up), and roll angle (positive right fin down) of the store. Of these six variables, axial position and roll angle do not generally require analysis. Change in axial position is a result of aerodynamic drag and rarely causes much movement in the half second after release. Roll angle can be quite large if the ejector is not centered directly on top of the store. Historically, this lack of alignment in ejector force over the center-of-gravity of the store has been difficult to quantify in the flight test community.

The first task was to compare the trajectory of the store released from an aircraft in the clean configuration to a store trajectory from an aircraft with a pod attached. The ATFLIR pod was chosen for this analysis since more flight test data was available for this pod than either of the other two. Large-scale wind tunnel data were used with NAVSEP in order to calculate the trajectory of the clean configuration, while the trajectory with the ATFLIR pod was predicted using wind tunnel data augmented with CFD results. The resulting displacements in the horizontal and vertical directions are shown below in Figure 35.

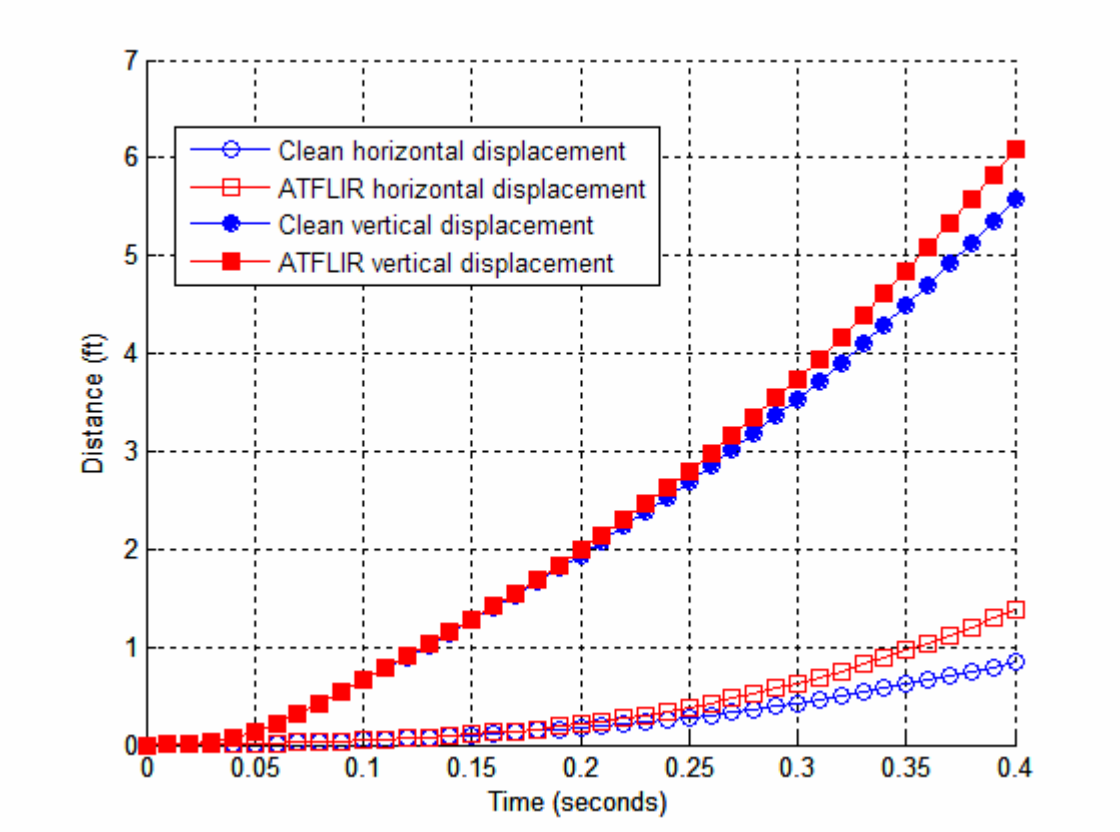


Figure 35. Predicted horizontal and vertical displacements for clean and ATFLIR cases

As this plot shows, the addition of the ATFLIR pod was predicted to cause a slight increase in the downward displacement and a 40% increase in outboard movement. In clean configuration, the store was predicted to move 0.90 feet outboard over the first 0.4 seconds after release, while with the ATFLIR attached it was expected to move 1.4 feet outboard. Moving farther outboard is not necessarily beneficial as it implies a greater yaw angle which could cause an impact of the store aft section with the ATFLIR pod. The variation in downward displacement was less than half a foot, differing between 6.1 and 5.5 feet after 0.4 seconds. This quantity is dominated by the force of the ejector so it is logical that variation would be minimal in this area.

In addition to displacements, pitch and yaw angles were also predicted for both the clean and ATFLIR cases. These results are plotted below in Figure 36.

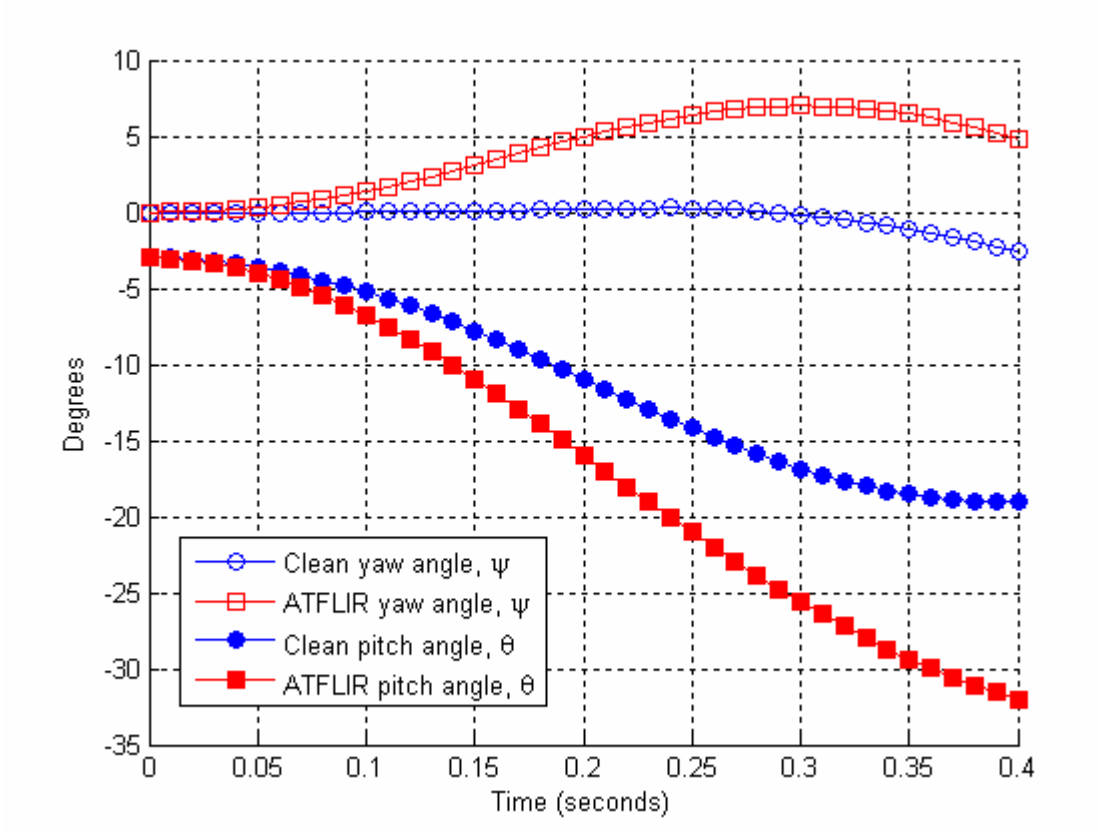


Figure 36. Predicted pitch and yaw angles for clean and ATFLIR cases

As this figure shows, the addition of the ATFLIR pod was predicted to increase both the yaw and pitch angles by a substantial amount over the clean values. The clean case was not expected to yaw at all, while the ATFLIR case was predicted to reach a yaw angle of 7.5 degrees outboard at 0.30 seconds after release. This is a substantial variation and is consistent with the pressure analysis discussed in the previous section. The pitching moment variation is also significant. The ATFLIR was predicted to cause a 68% increase over the clean value, increasing the nose-down angle from 19º to 32º 0.4 seconds after release. Such a large pitch angle can be worse than a large yawing angle, as the store can rotate so quickly that the tail moves upward and strikes the pylon. This is a particular concern with large stores such as this one, as the tail moves a greater distance upward with the same change in pitch angle.

The comparisons between the predicted clean and ATFLIR trajectories correlate with the trends that were predicted by the analysis of forces and moments on the store in the carriage position. In order to validate these data it was necessary to compare the predicted trajectories

with actual flight test results. Therefore trajectories were compared for both the clean and ATFLIR cases using vertical position, horizontal position, yaw angle, and pitch angle. The plots for vertical and horizontal position can be seen below in Figure 37 and Figure 38.

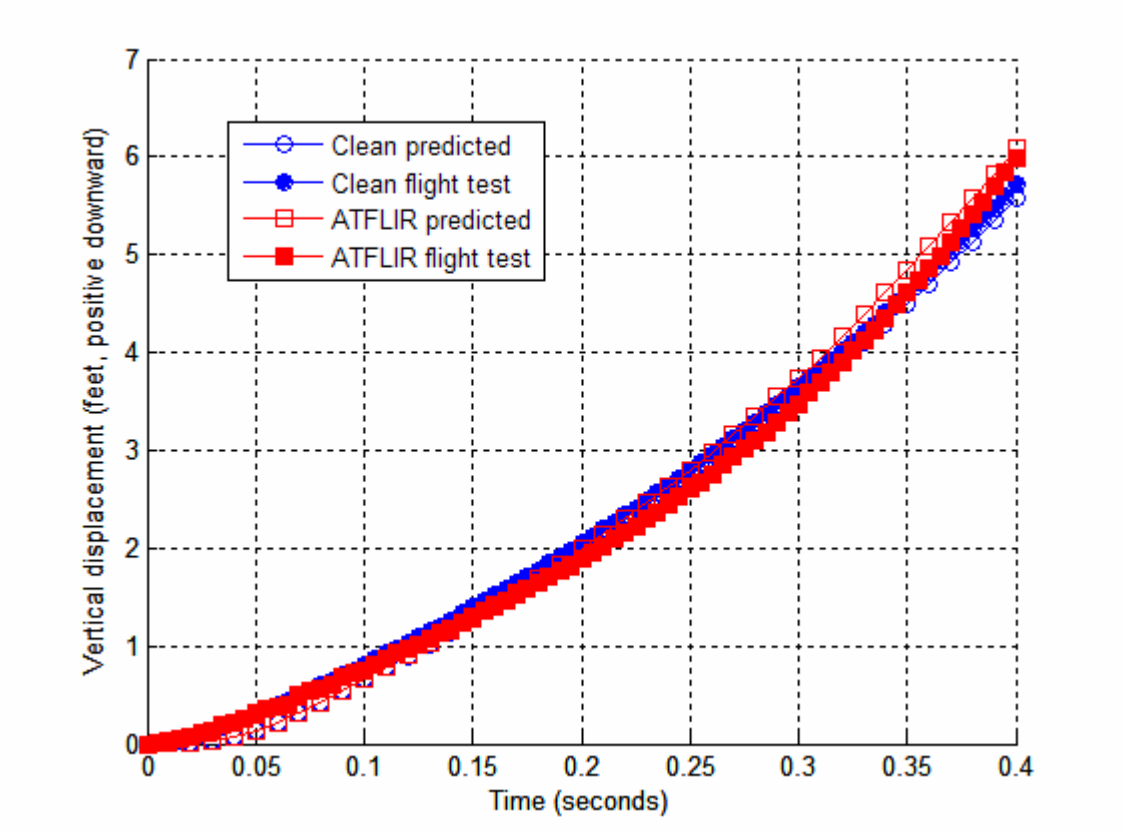


Figure 37. Vertical position flight test comparison

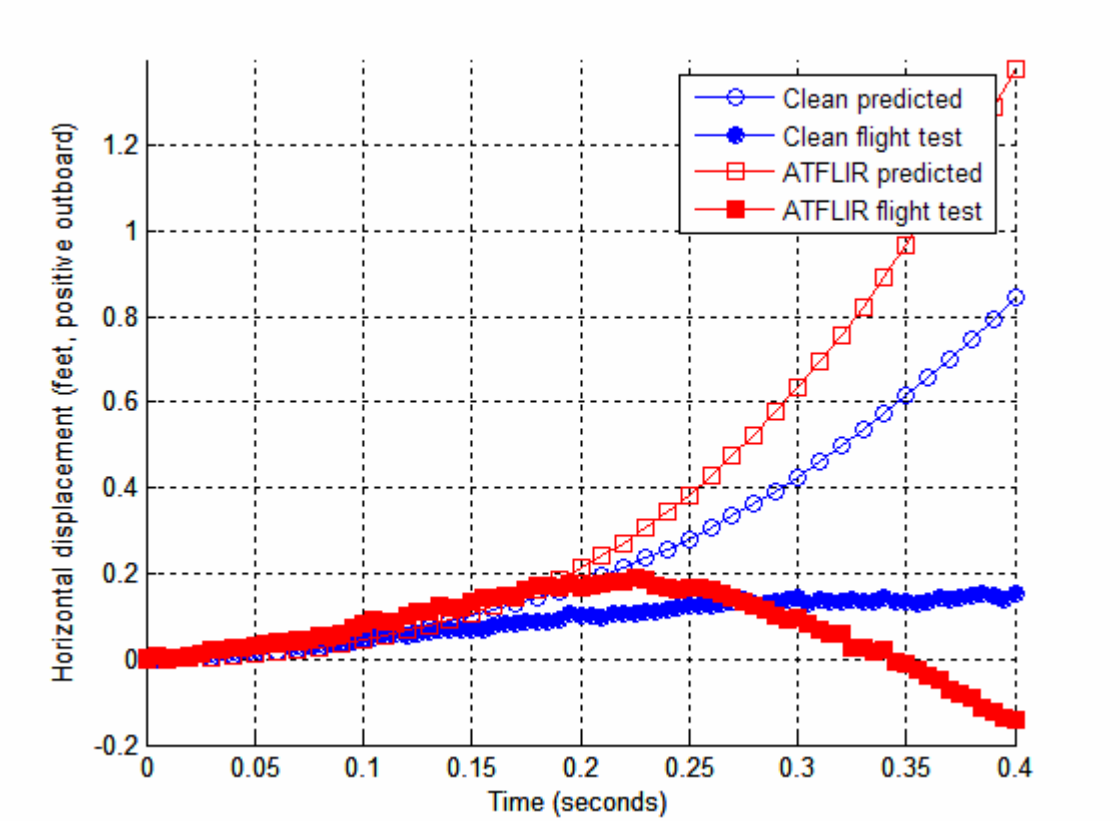


Figure 38. Horizontal position flight test comparison

As the first plot shows, the predicted vertical position results match almost perfectly with flight test data, indicating a good model of the ejector system. Results in horizontal displacement diverged at the later time portion but did match well for the first 0.2 seconds. The variation is much greater between the predicted and actual ATFLIR results than it is between the predicted and actual clean results. This difference is likely due to the manner in which the CFD data was used to augment the wind tunnel data, which provided the most accuracy near the carriage position and became less accurate as the store moved farther away. This would explain why the results match well initially and diverged at later times. Another possible explanation for the difference in spatial displacements is associated with the manner in which the data is captured from flight test. In this case, the flight test data came from photogrammetrics. The camera images used as the basis for photogrammetric calculations are obtained from cameras primarily situated abeam the delivery aircraft. Depth in the frame of the camera is associated with y-displacement and has a higher degree of uncertainty than do the vertical and axial

displacements. The magnitude and direction of the movement in the horizontal direction are so small, though, that this variation is not overly critical. The worst case flight result showed the ATFLIR moving less than 3.0 inches towards the aircraft after 0.4 seconds, by which time the store has already moved 6.0 feet downward.

The other two plots generated in this section compare the yaw and pitch angles between the predicted and flight test cases. These plots can be seen below in Figure 39 and Figure 40.

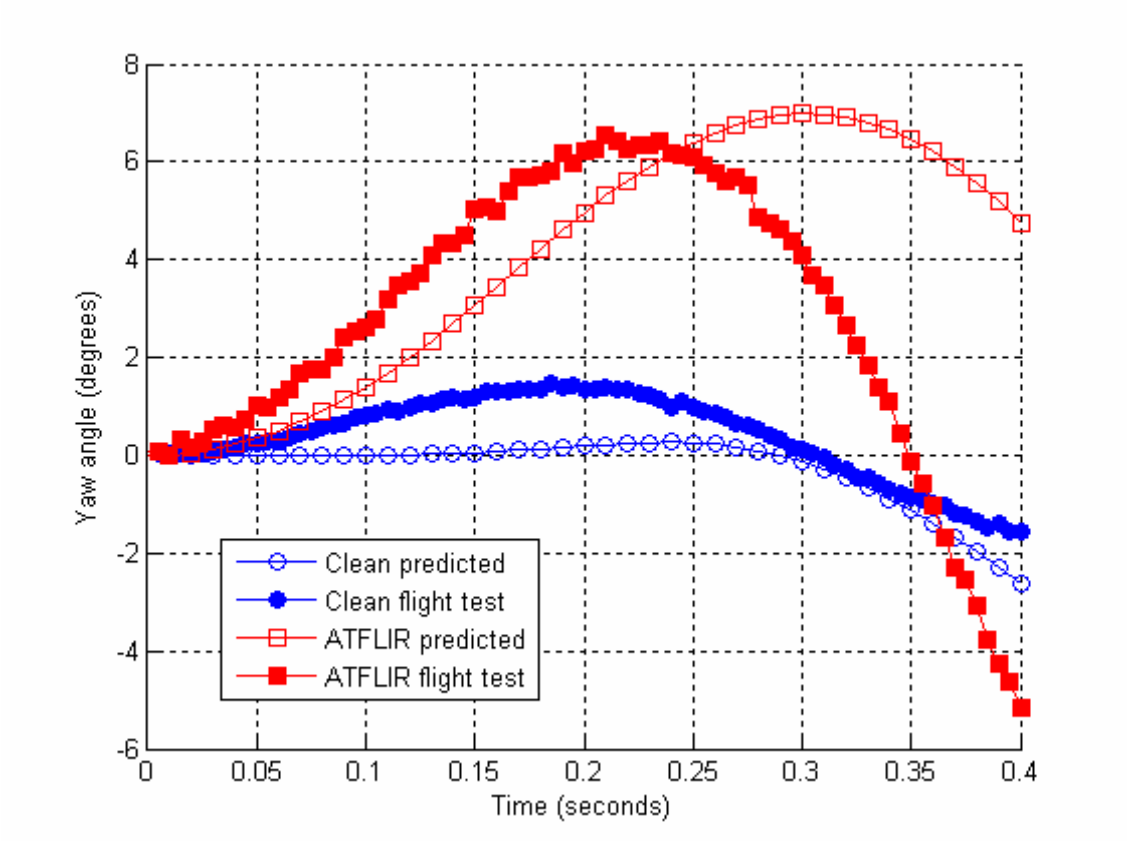


Figure 39. Yaw angle flight test comparison

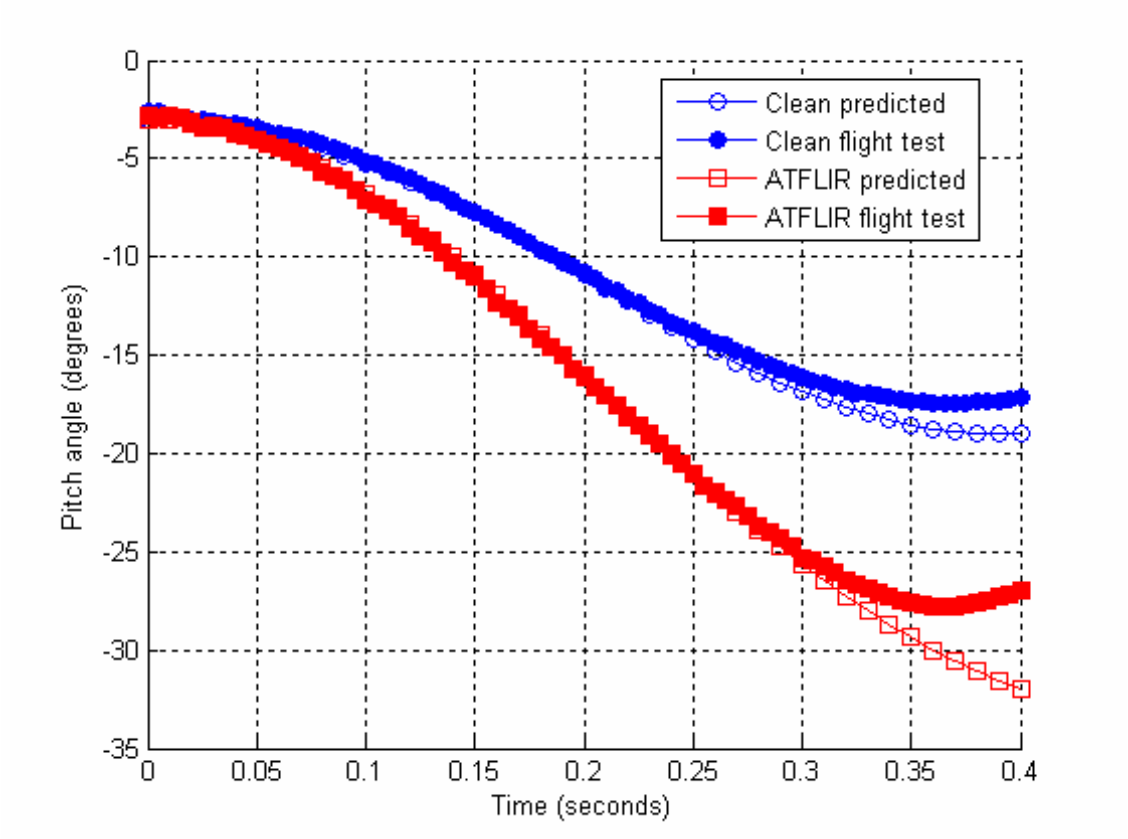


Figure 40. Pitch angle flight test comparison

Both of these plots reveal good correlation between the predicted trajectory and flight test results. The predicted maximum yaw angle is a mere 0.3° from the actual one, with the predicted value showing the slightly higher value of 7.0°. The correlation of the predicted clean trajectory was good as well, never varying by more than two degrees. The greatest discrepancy on this plot is the variation of the ATFLIR results at later times. This difference is possibly due to errors in the method used to calculate the flight test results. Even so, it is noteworthy that the prediction exhibited the same trends and maximum as the flight test result. At lower times it is also important that the prediction was a few degrees lower in both the clean and ATFLIR cases, which shows that that prediction accurately captured the differences between the cases in carriage position. The comparison of pitch angle in Figure 40 is an even better match than yaw angle. As this plot shows, the predicted values were exact matches for flight test results for the

first 0.3 seconds after release. Such a change in pitch angle can cause significant problems for stores so it is important that this result was accurately captured.

Based on these comparisons of predicted and actual trajectories for both the clean and ATFLIR configurations, as well as previous validations of both the F-18C aircraft and GBU-31 JDAM, the data generated in this project were considered accurate and accepted for further use. After this point no new data were generated and the remainder of this research effort consisted of post-processing and analyzing the results.

4 ANALYSIS

While the results presented up to this point have demonstrated the applicability of CFD to store separation analysis, no effort has been made to explain the actual phenomenon causing the targeting pods to influence the forces and moments on the store. This is one area in which CFD can provide insight into aspects of the flowfield that cannot be obtained from either typical store separation wind tunnel data or flight testing. Flight tests can typically only be used to measure trajectories, which are very useful but do not yield any force or moment data. The recent use of miniature telemetry packages have greatly improved the accuracy of the flight test data, and can allow the inference of some force and moment data on the store. Unfortunately, flight test data of this type were not taken during the targeting pod flight test clearance effort. Wind tunnel testing allows for measurement of forces and moments and the velocity of the flow at any point using a probe, but the models are almost always scaled and time constrains the number of points which can be tested. Store separation testing at transonic speeds requires specialized tunnels capable of measuring the forces and moments on a store that can be moved in relation to the aircraft test article. These are termed CTS tunnels, and their availability is restricted. Two examples of such tunnels are the 16T at Arnold Engineering Center in Tennessee, and the 8 FT Transonic Tunnel at CALSPAN in Buffalo, NY. Wind tunnel techniques using flow visualization such as Pressure Sensitive Paint (PSP) or Particle Image Velocimetry (PIV) are not common for store separation testing in these tunnels.

CFD solutions do not have any of these limitations. By definition when a flow is "solved" using a CFD package, the velocity, pressure, density, and energy of the fluid are known at each discrete point in the flow corresponding to the nodes in the grid. Results at interim points can accurately be interpolated where the grid is sufficiently fine. In practice, this means that all of these variables are known at all points in the grid, and recall from Table 3 that there are nearly 2.5 million points in each of the carriage position grids. These data can be used for many different purposes, such as examining the pressure distribution along the surface of a store or examining any flow variable along a given plane, among many others. The computational limitation of CFD for this purpose is that a separate solution is required for each position of the

store, while in the wind tunnel the model of the store can simply be moved and more data taken nearly immediately.

Specialized flow visualization programs are needed to analyze such large amounts of raw data, and in this research project Amtec Engineering's Tecplot® was used for this purpose. Tecplot® is an advanced graphics program designed for many types of engineering data. Important features pertinent to this project include the ability to display pressure surface pressure contours and calculate the location of shocks in the flow.

4.1 Shock Analysis

All store separation issues involving these pods occurred in the transonic flight regime, making the formation of shocks a likely aspect of the problem. The transonic flight regime is defined as the range between Mach 0.80 and 1.20, and at these speeds it is very common for numerous shocks to form at different locations on the aircraft. Shocks form because supersonic flow cannot change direction in the same manner as slower, subsonic flow. In order to change direction, supersonic flow requires either a shock wave or an expansion wave. Shock waves generally cause more drag and create greater changes in the flow's density and pressure, and are therefore usually more important to analyze. When a shock occurs, the flow downstream of the shock moves at a slower speed and has a higher density, temperature, and pressure than the flow ahead of the shock. This aspect makes shock analysis very important to this project, as a shock impacting the store could have a dramatic effect on the store's surface pressure distribution. Unfortunately, while shocks can be calculated analytically over basic configurations they are very difficult to predict over complex geometries such as the one analyzed here, making CFD an ideal tool for this investigation.

Figure 41 below shows the underside of an F-18C Hornet with an ATFLIR pod attached and a GBU-31 JDAM in the carriage position at Mach 0.92. The data have been mirrored so the left and right sides of the aircraft are images of one another, which allows for more perspective on the same image.

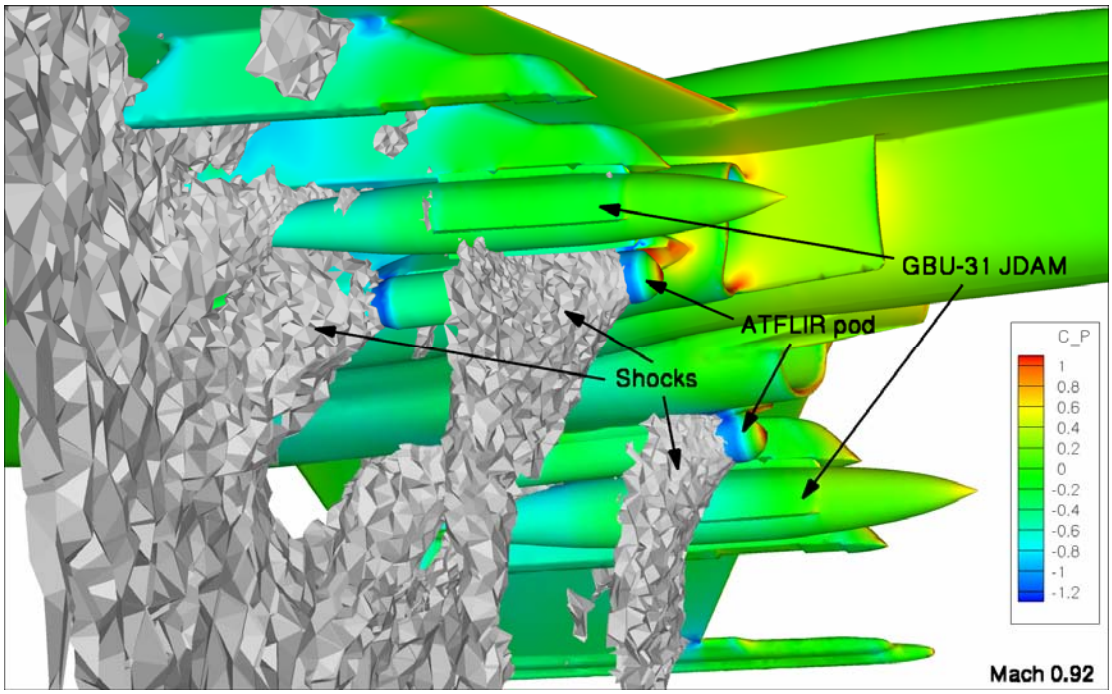


Figure 41. Shock analysis with ATFLIR pod at Mach 0.92

The colors on the surface of the aircraft and store represent the pressure distribution using the pressure coefficient explained by Equation 4. Colors closer to red indicate higher pressure while those closer to blue indicate lower pressure, as the legend shows on the right side of the figure. The shock surfaces, labeled and colored in gray, were calculated using a flow analysis tool in Tecplot®. They appear faceted due to the size of the grid cells at each point, but in reality would be smooth.

An examination of this figure shows that there are two distinct shocks forming on the ATFLIR pod, one near the nose of the pod and one at its trailing edge. The two shorter arrows from the "Shocks" label point to the one at the leading edge, while the longer arrow points to the shock formed at the trailing edge. Both of these shocks impact the GBU-31 JDAM in its carriage position. The leading shock impacts the bomb near its center, while the trailing shock impacts near the fins. Recall that initial speculation centered on the difference in leading edge geometry between the ATFLIR and TFLIR as the likely explanation for their different effects on store trajectory. As a result of this hypothesis, the leading edge shock was examined first. While this leading edge shock did cause a change in pressure where it impacted the store, it was soon found to remain relatively consistent with changing Mach number. A visual examination of this

shock at various Mach numbers showed that it did not move or change the surface pressures enough to cause the drastic change in store yawing moment as predicted by CFD results.

In order to determine if the second shock moved significantly over the range of transonic Mach numbers, images of the shocks on the underside of the aircraft were made from the same perspective at each Mach number. The aircraft was again mirrored, though the shock was only displayed on the left side of the aircraft to allow for better viewing of the pressure distribution on the right. Figures 42-44 show these images at Mach 0.85, 0.90, and 0.95.

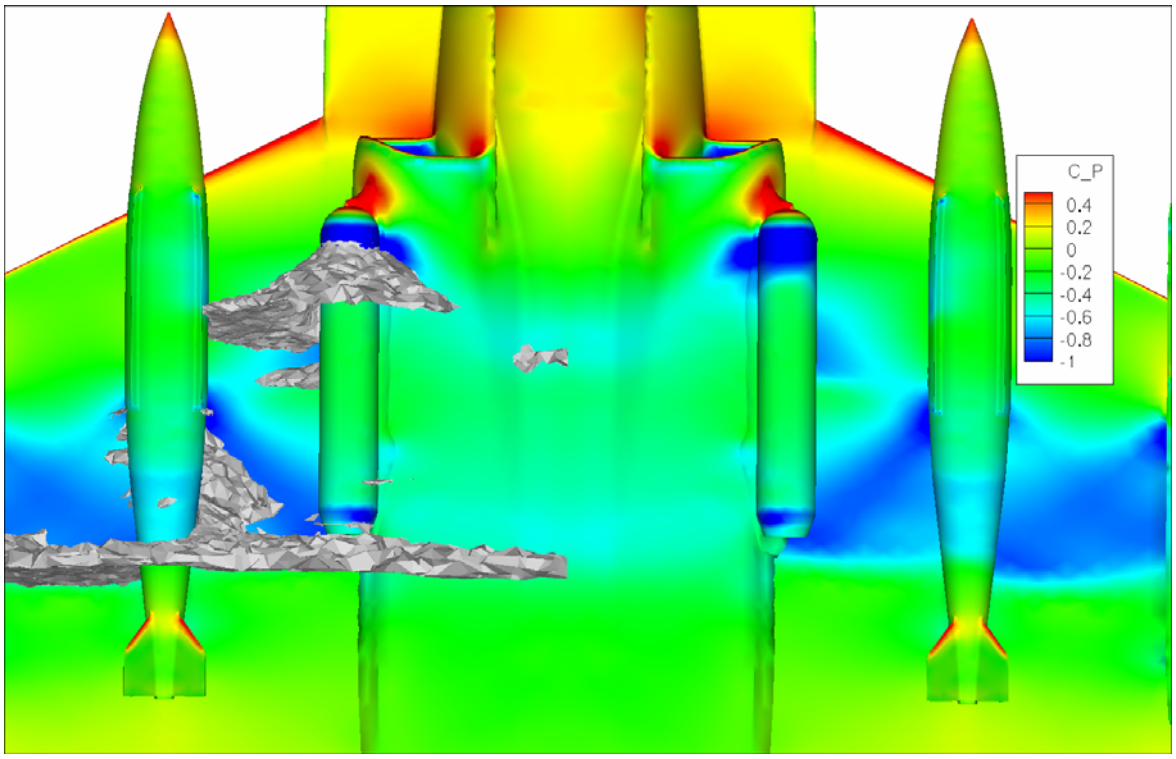


Figure 42. Shock formation on F-18C underbelly with ATFLIR pod at Mach 0.85

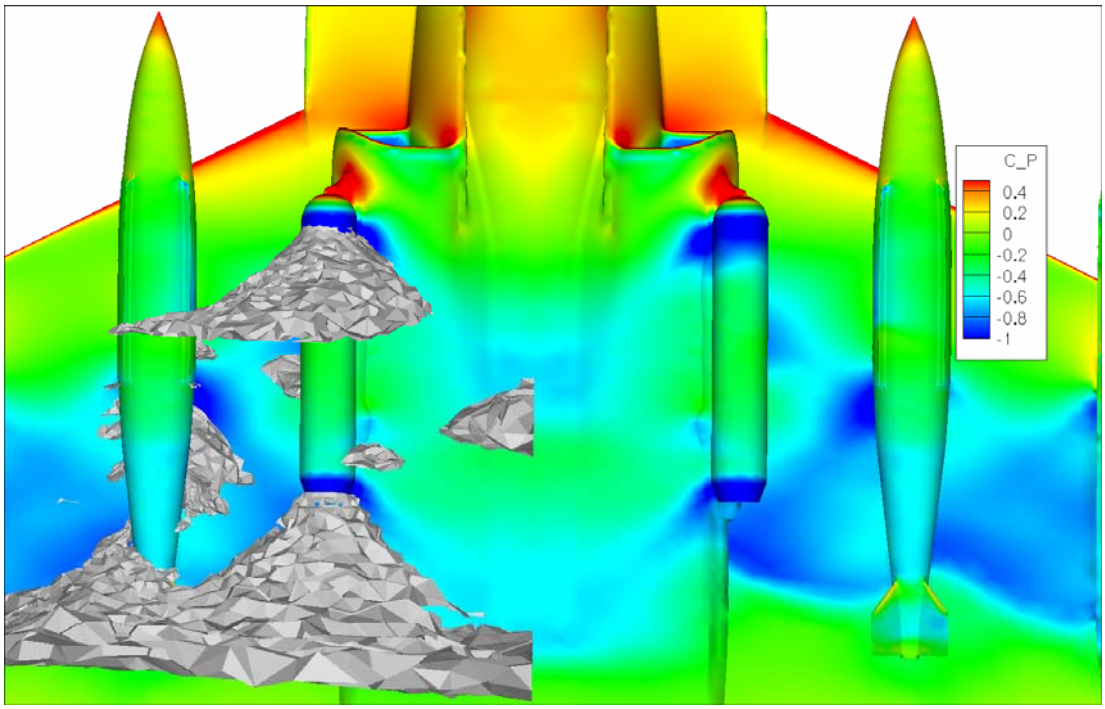


Figure 43. Shock formation on F-18C underbelly with ATFLIR pod at Mach 0.90

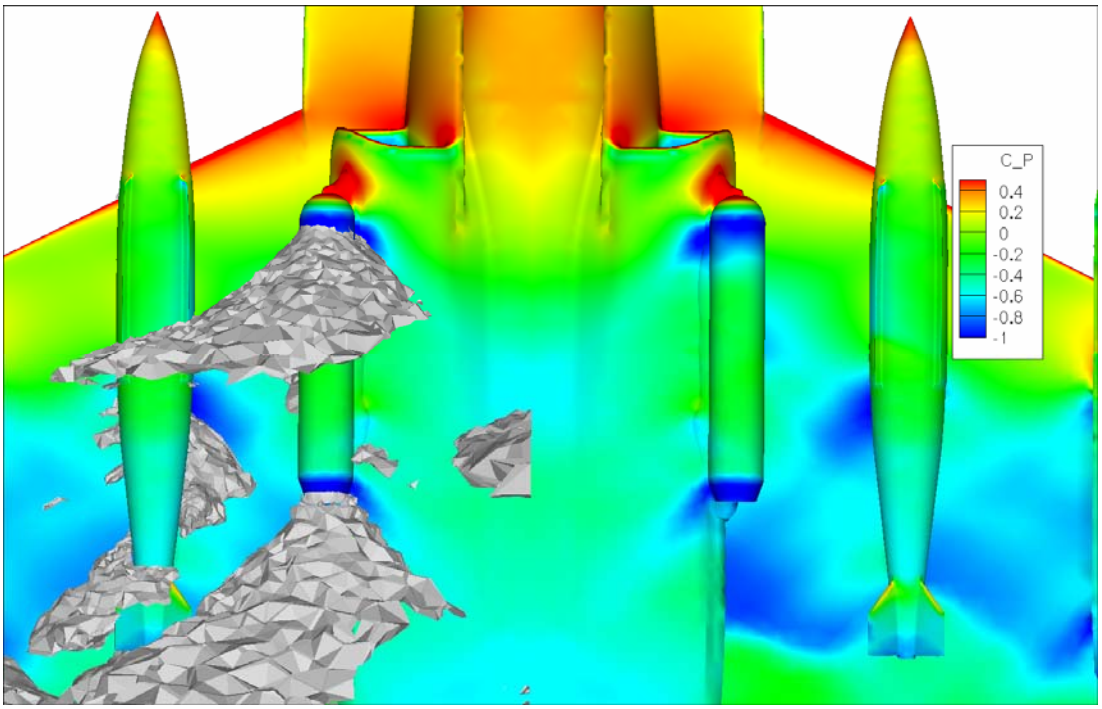


Figure 44. Shock formation on F-18C underbelly with ATFLIR pod at Mach 0.95

As these images show, the shock produced by the trailing edge of the ATFLIR pod does move a significant amount over this range of Mach numbers. The shock is barely formed at Mach 0.85, impacts the tail at 0.90, and aft of the tail by 0.95. By contrast, the shock produced by the leading edge impacts the GBU-31 JDAM at the same location regardless of speed. While these images do not give any quantitative data, they strongly suggest that the trailing shock is the cause of the pressure changes.

In order to ensure that these shocks only formed due to the presence of the ATFLIR pod, images from the same perspective and at the same flight conditions were made for the case with no targeting pod. These can be seen below in Figures 45-47.

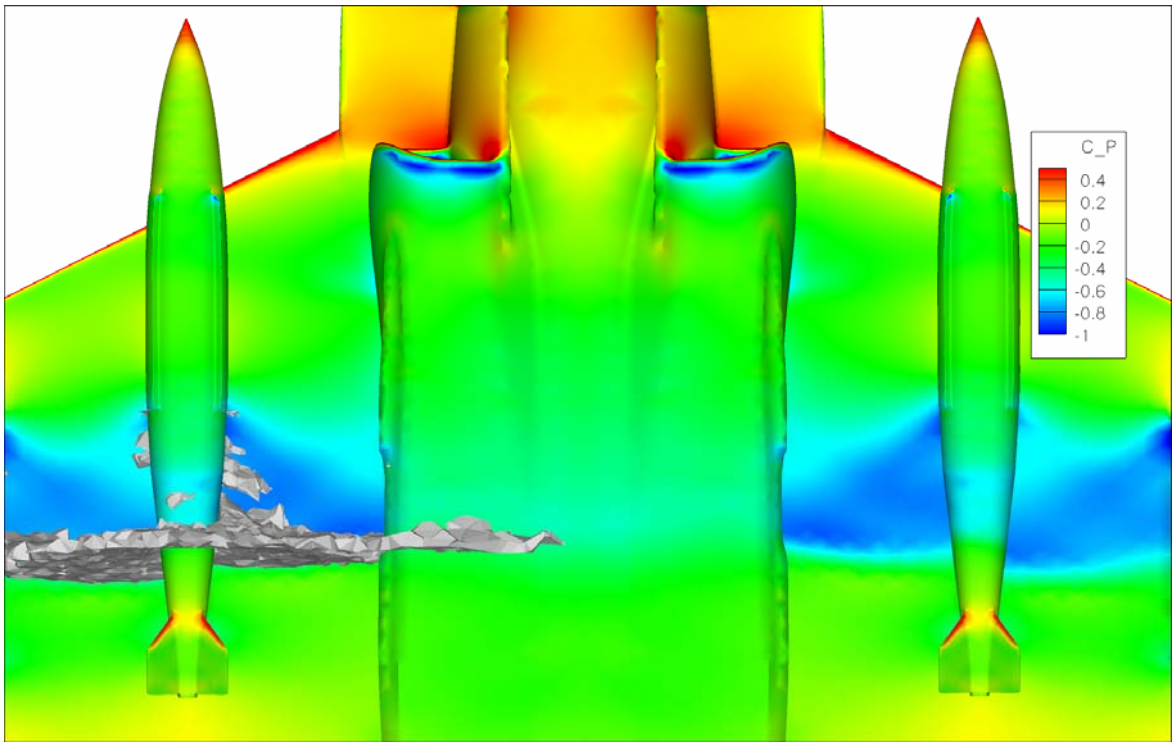


Figure 45. Shock formation on F-18C underbelly at Mach 0.85 in clean configuration

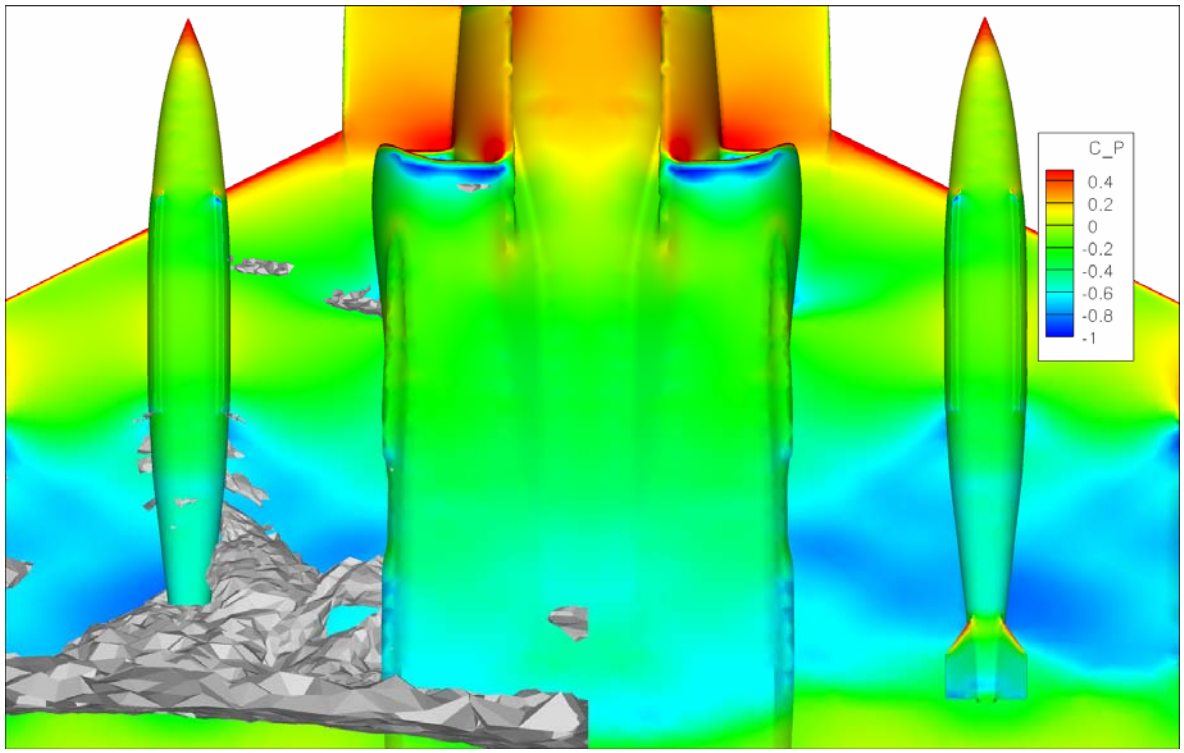


Figure 46. Shock formation on F-18C underbelly at Mach 0.90 in clean configuration

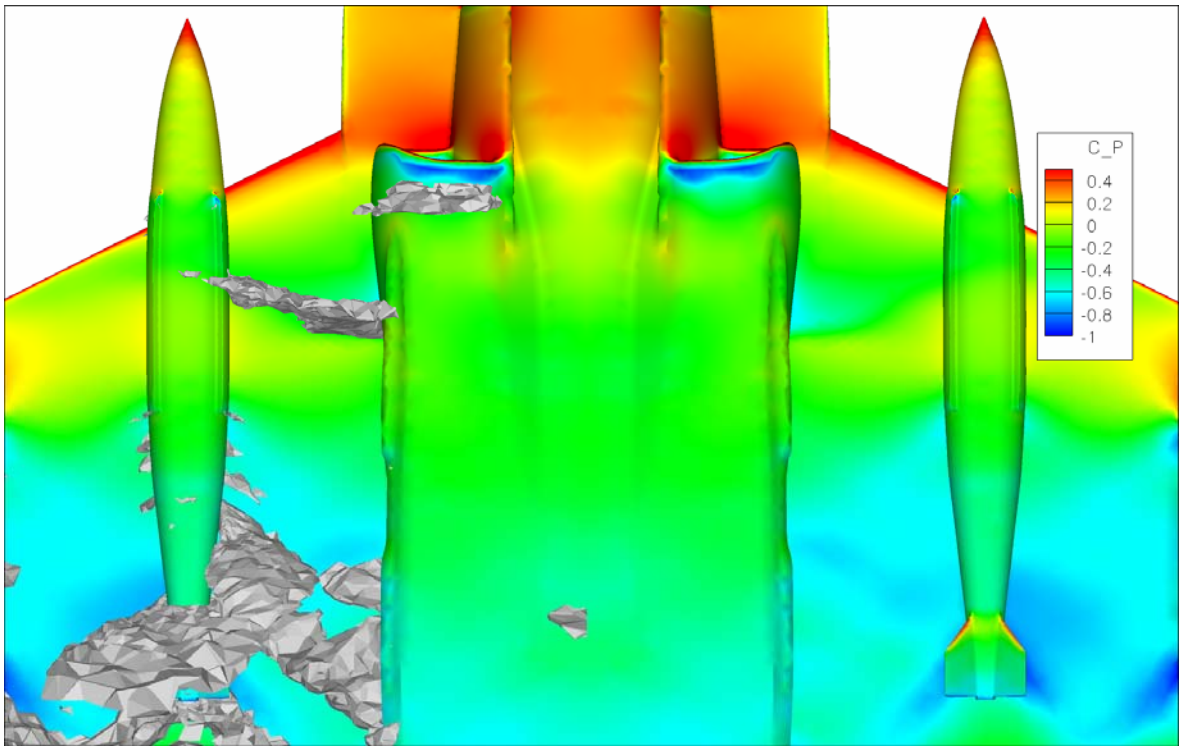


Figure 47. Shock formation on F-18C underbelly at Mach 0.95 in clean configuration

As these plots show, there is a shock that moves aft in the clean configuration. This shock, however, is nearly symmetric and does not cause as intense a change in surface pressures as the shock from the ATFLIR does. In the case of the ATFLIR, there is a distinct line on the underside of the wing where low pressure became high pressure. This transition is much more gradual at higher speeds in the clean configuration.

4.2 Component Analysis

Based on the observations made when analyzing shock formation, it was likely that a shock impacting the tail caused the variation in store yawing moment. In order to prove whether or not this was the case, the GBU-31 JDAM was divided into three regions and each was analyzed separately. The regions analyzed were the forebody, midsection, and tail, as shown in Figure 48 below.

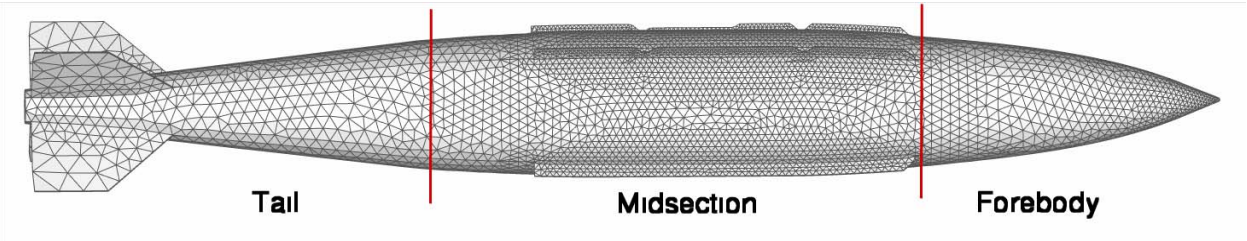


Figure 48. JDAM component locations

Version 6.0 of the flow solver USM3D allowed for the integration of pressures on multiple components, so this feature was used to determine the contribution made by each component to the store's total forces and moments. Figures 49 and 50 show the contributions to yawing moment made by the forebody and midsection, respectively. While both sections of the store show a higher yawing moment produced by the cases with pods attached, the magnitude of the differences is quite low. The largest change on the forebody is a difference of 0.20 between the Litening pod and the clean configuration. This variation is also constant across all Mach numbers, so the forebody has no net effect on the total yawing moment spike at Mach 0.90.

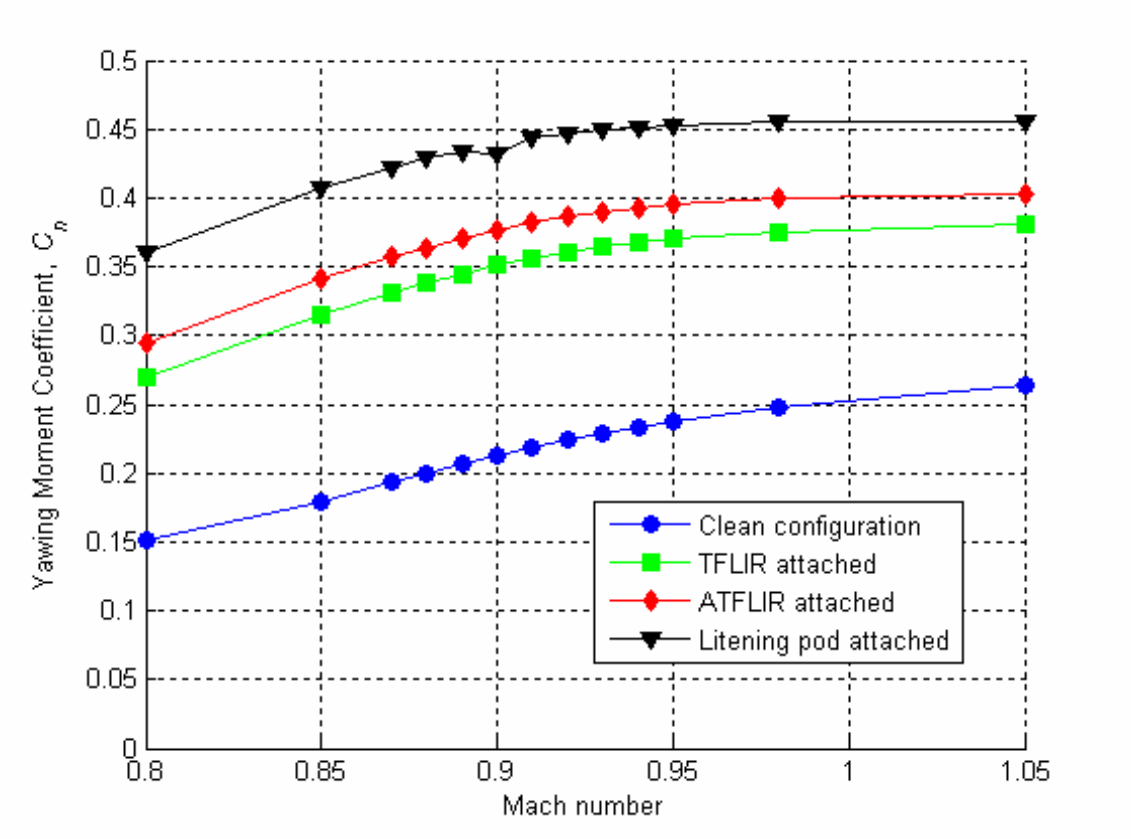


Figure 49. GBU-31 JDAM forebody yawing moment component

The midsection of the GBU-31 JDAM was shown to be nearly as insignificant as the forebody in terms of yawing moment. Figure 50 below shows the variation in this quantity, and as similar to the forebody the greatest variation was only 0.30. This result proved that the shock formed on the leading edge of the ATFLIR pod did not have an effect on the yawing moment variation. This conclusion was drawn from an examination of shock positioning and reinforced numerically with these results.

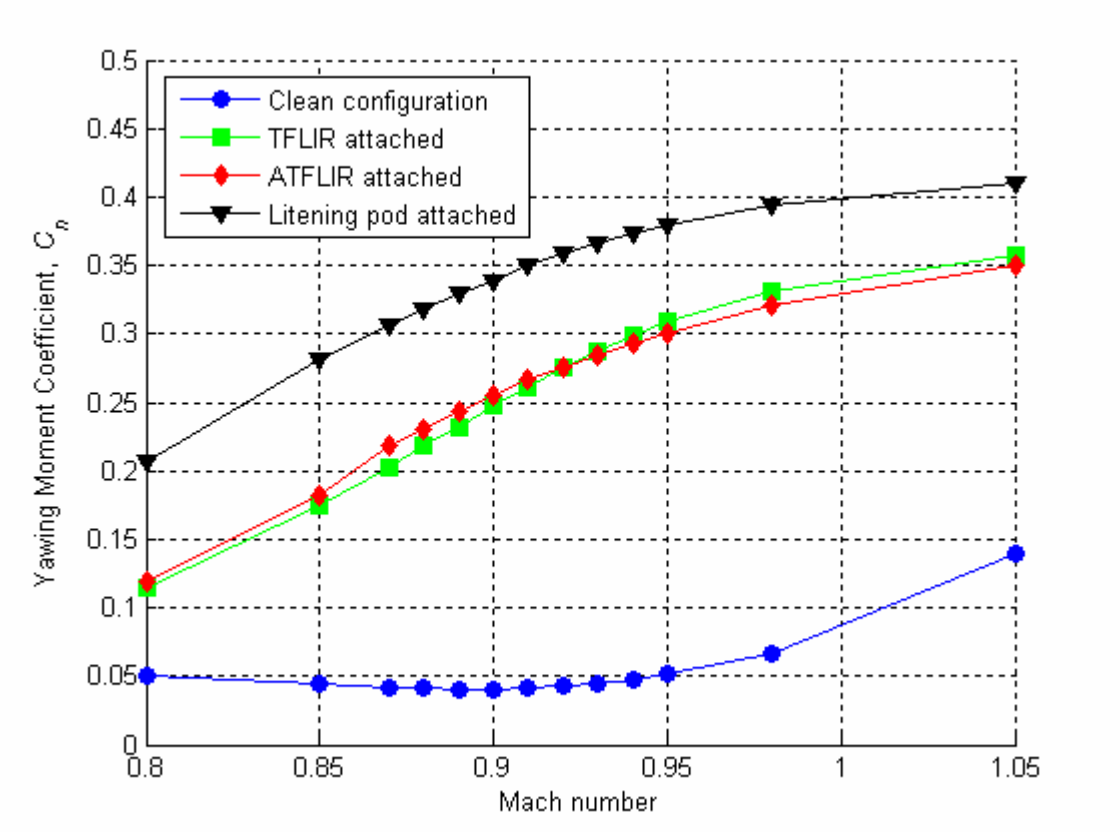


Figure 50. GBU-31 JDAM midsection yawing moment component

If both the forebody and midsection of the GBU-31 JDAM showed minimal change in yawing moment as Mach number increased, the tail had to be the important feature in this regard. Figure 51 below shows that this was the case. In clean configuration, the yawing moment caused by the tail remained relatively constant, varying by only 0.40. This configuration peaked at Mach 0.89, which is near the speed where the pod configurations minimized.

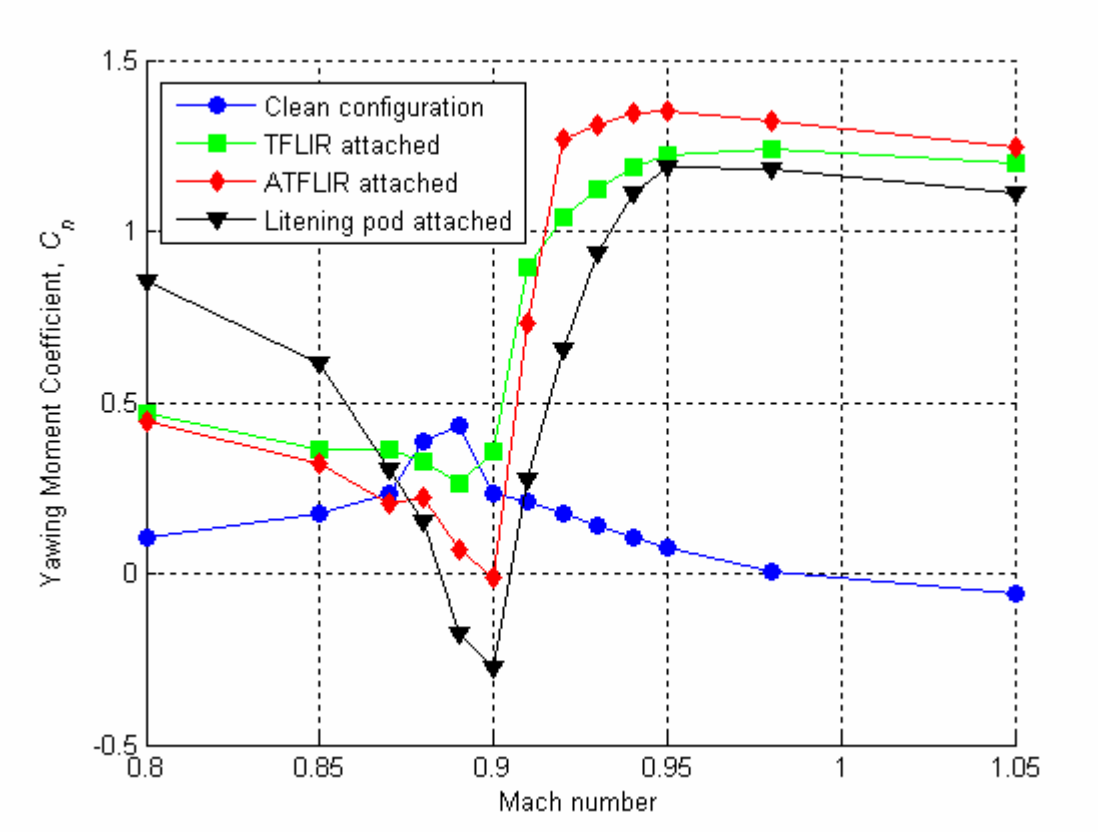


Figure 51. GBU-31 JDAM tail yawing moment component

As this plot reveals, the tail was clearly the most critical portion of the store. It exhibited dramatic changes in yawing moment, most especially from Mach 0.90 to 0.92. Recall from the images in Figure 42 through Figure 44 that this is the same speed range when the trailing edge shock was sweeping across the tail. At this point it could be concluded that the trailing edge shock was the cause of the shift in yawing moment, but just to be sure images of the pressure distributions on the inboard and outboard sides of the bomb's tail were generated at Mach numbers in this region. The range of Mach 0.90 to 0.92 can be seen in Figure 52 through Figure 54 below.

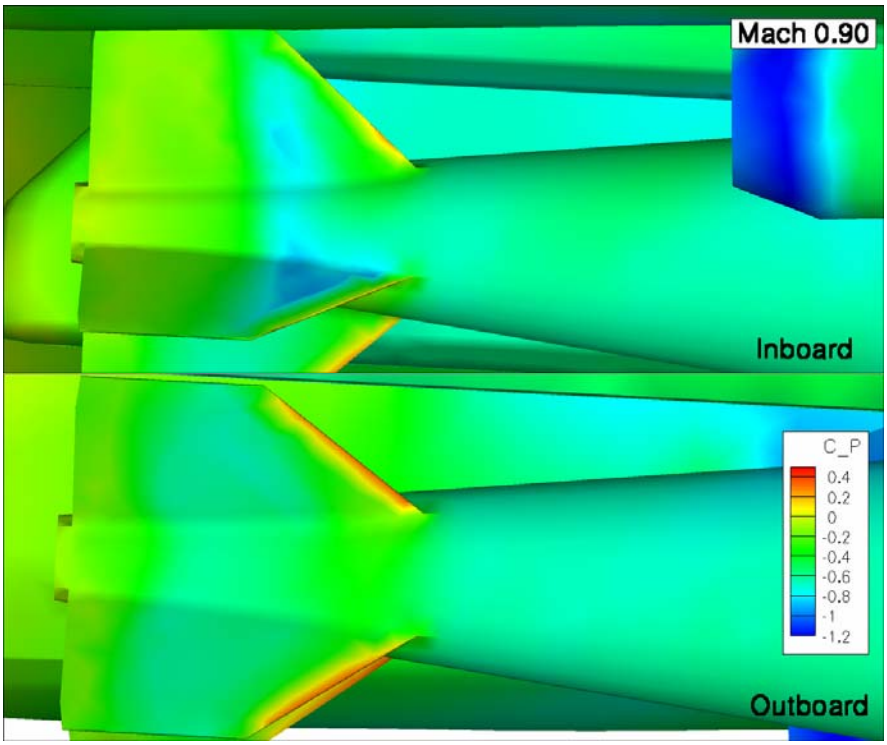


Figure 52. GBU-31 JDAM tail pressure distribution at Mach 0.90

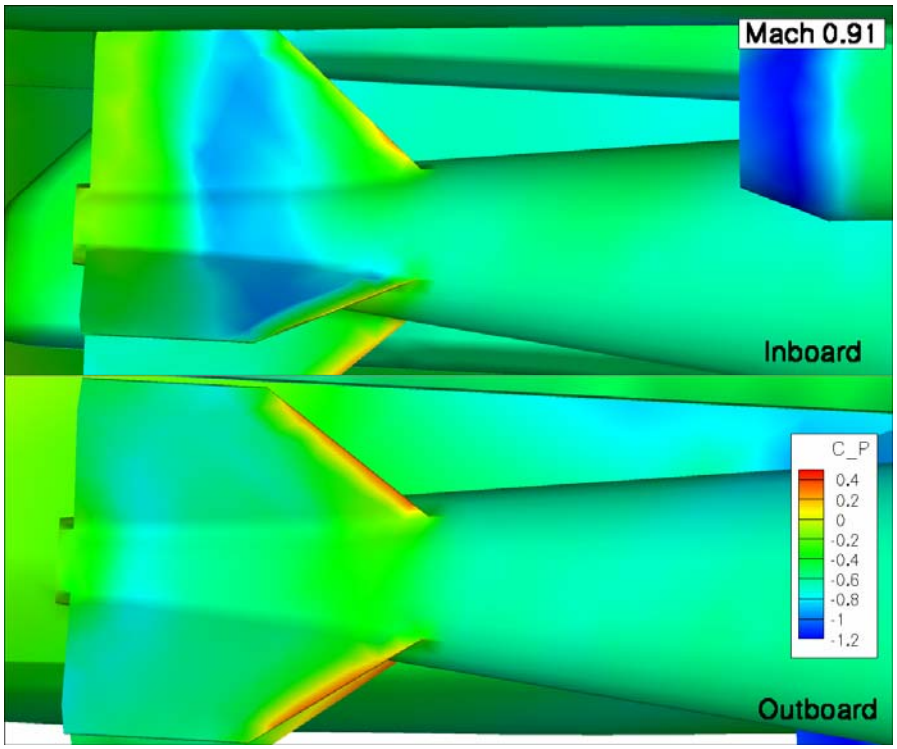


Figure 53. GBU-31 JDAM tail pressure distribution at Mach 0.91

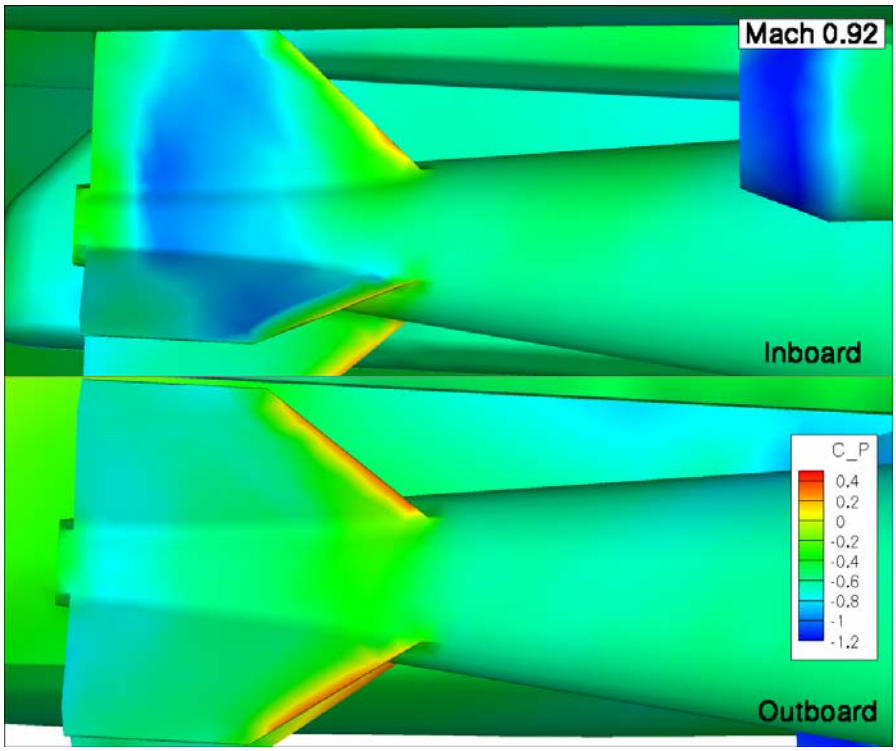


Figure 54. GBU-31 JDAM tail pressure distribution at Mach 0.92

As these figures show, the pressure on the outboard side of the tail remains relatively constant while the inboard side changes quite a bit. At Mach 0.90 the majority of the inboard section of the tail is at a relatively high pressure (in green and yellow), representing a pressure coefficient value of around 0.0. There is only a small region of low pressure (in blue) on the front portion of the fins, which represents a much lower pressure coefficient of -1.0. A variation of 1.0 in pressure coefficient is a very significant difference, indicating that the region of lower static pressure is experiencing nearly double the dynamic pressure of the region in greenish-yellow. The division between the high and low pressure regions moves aft as speed increases, which significantly decreases the overall static pressure on the inboard side of the store. The pressure on the outboard side of the tail remains relatively constant through this speed range, so the decrease in pressure on the inboard side causes a pressure imbalance which results in a net aerodynamic force pushing the tail inboard. This force is therefore the cause of the store yawing moment change shown to occur over this range of velocities. The cause of the division between areas of low and high pressure on the tail's inboard side is the impact of the shock generated on the trailing edge of the ATFLIR pod. As stated previously, an important property of shocks is

that they cause an increase in static pressure when they occur. A shock is in fact the only explanation for such a dramatic increase in pressure along flat surfaces such as these tail fins. Another property of shocks is that they are swept further back at higher velocities, which can be seen in image form in Figure 42 through Figure 44. The sweeping of the ATFLIR's trailing shock is the reason that the area of lower pressure on the tail continued to grow until the entire inboard side was at the lower pressure. These images are final confirmation that the trailing edge shock from the ATFLIR was the cause of the store's drastic increase in yawing moment.

5 CONCLUSIONS

The solution of complex flow problems using computational fluid dynamics is increasingly being used to augment traditional methods of aerodynamic testing, such as the use of wind tunnel and flight tests. This is especially true in the field of store separation, where the past two decades have seen CFD make significant contributions to the field in many areas. CFD has been particularly useful in store separation applications that simply cannot be replicated in the wind tunnel or can only be replicated with great difficulty. An example of this includes such applications as the release of a store from a cavity such as a bomb bay of a B-1B aircraft. CFD has also proved to be useful in the visualization of complex flows in order to better understand the elements that affect such flows. This research has demonstrated the use of CFD as a flow visualization tool to capture a complex shock interaction between a store and an element of the F-18C aircraft geometry, namely the targeting pod. In addition to identification of the changes in aircraft geometry resulting in adverse store separation conditions, CFD was also able to quantify those adverse effects. The quantified effects then served as input to a trajectory simulation program which used the best available data from ejector force data, wind tunnel testing, and CFD to predict the motion of the store after release. The predicted motion of the store and quantified effects of the transonic flow on the store correlated well with the observed behavior of the shock interference in the flow. This led to a number of recommendations concerning the trailing-edge geometry of the targeting pod on the F-18C that could potentially result in significant increases in combat effectiveness.

CFD is, however, not an experimental method but an approximate solution to a set of equations which have no known analytic solution. As such, the process of validation of each step of a CFD investigation is as important as the results themselves. CFD results without proper verification and validation lack the pedigree to provide the store separation engineer with the confidence to make flight clearance decisions. Therefore each step in the analysis of the store separation simulation using CFD was verified against existing wind tunnel and flight test data. Excellent agreement was found for the freestream flow characteristics of the GBU-XX store using CFD to large scale wind tunnel testing in two separate wind tunnels. Flow quality of the

CFD solution around the F-18C aircraft itself was verified at the carriage location of the store against existing wind tunnel data, and engine parameters in the CFD model were tuned appropriately. Finally, the predicted trajectory of the store after release using results from CFD was validated against flight test data for the release of the store from a similar condition. Good agreement was found in each case, giving the store separation engineer the confidence in both the predicted trajectory of the store and the rationale for the adverse deviation of the predicted trajectory from nominal when a targeting pod is attached to the aircraft.

This research effort has proven the efficacy of computational fluid dynamics to the field of store separation analysis. CFD methods were used to accurately model complex aircraft geometries and generate results consistent with those produced by the wind tunnel and flight testing. The analysis of subtle geometric changes was quantified, which enabled the determination of the specific aspect of the geometry which caused the detrimental effect. A more thorough, accurate, and insightful understanding of the store separation environment of the F18C carrying a targeting pod in the transonic flow regime was obtained.

5.1 Future Work

This research will be continued at the High Performance Computing center and will focus on the effect of geometric modifications to the trailing end of the targeting pods. It is expected that a smoother trailing end will weaken shock formation and thus mitigate the adverse conditions currently experienced. Additional research could also refine the trajectory predictions using CFD data in lieu of wind tunnel data employed in this effort.

5.2 Recommendations

The capabilities of CFD have matured to the point that it is an integral part of the store separation analysis process. The cost of CFD analyses is substantially lower than both wind tunnel and flight tests, so using CFD as the initial investigatory tool makes good fiscal sense. Certain conditions where CFD results indicate dangerous situations can then be selected for further testing in the wind tunnel or via flight test. Promising changes in geometry or location as identified from CFD analysis of additional weapon systems, such as the ATFLIR, should guide the test plan for further wind tunnel and flight testing.

BIBLIOGRAPHY

Anderson, John D., Computational Fluid Dynamics: The Basics with Applications. New York: McGraw-Hill, Inc., 1995.

Benmeddour, A., Mébarki, Y., Orchard, D., Tang, F.C., and Fortin, F., "Computational Investigation of the FLIR Pod Interference Effects on the CF-18 Aircraft Stores," ICAS paper 684, Sept. 2006.

Cenko, A. and Lutton, M., "ACFD Applications to Store Separation – Status Report," ICAS Paper 231, 2000.

Cenko, A., "Configuration Effects on the F-18 Aircraft Flowfield." NAVAIR Report No. NADC-90111-60, May 7, 1990.

Cenko, A., Niewoehner, R., and Rychebusch, C., "Evaluation of the Capabilities of CFD to Predict Store Trajectories from Attack Aircraft," ICAS paper 240, 2002.

Cenko, A., "Experience in the Use of Computational Aerodynamics to Predict Store Release Characteristics," *Progress in Aerospace Sciences*, Vol. 37, 2001, pp. 477-495.

Cenko, A., "F-18/JDAM CFD Challenge Wind Tunnel Flight Test Results," AIAA Paper 99-0120, Jan. 1999.

Chaffin, M.S., and Pirzadeh, S., "Unstructured Navier-Stokes High-Lift Computations on a Trapezoidal Wing," AIAA Paper 99-1191, 1999.

Frink, N., Pirzadeh, S., and Parikh, P., "The NASA Tetrahedral Unstructured Software System (TetrUSS)," ICAS Paper 0241, August 2000.

Frink, N., "Recent Progress Toward a Three-Dimensional Unstructured Navier-Stokes Flow Solver," AIAA Paper 94-0061, January 1994.

Frink, N., and Pirzadeh, S., "Tetrahedral Finite-Volume Solutions to the Navier-Stokes Equations on Complex Configurations," Tenth International Conference on Finite Elements in Fluids, January 1998.

Frink, N., "Tetrahedral Unstructured Navier-Stokes Method for Turbulent Flows," AIAA Journal, Vol. 36, No. 11, November 1998, pp. 1975-1982.

Frink, Neal, et. al., "TetrUSS," http://aaac.larc.nasa.gov/tsab/tetruss/mac/TetrUSS-2005-0812.pdf. October 2006, (October 2007).

"Joint Direct Attack Munition," http://en.wikipedia.org/wiki/Image:GBU-31_xxl.jpg#filehistory, (November 25, 2007).

Mani, M., Cary, A., and Ramakrishnan, S.V., "A General Purpose Euler and Navier-Stokes Solver for Structured and Unstructured Grids," *Journal of Aircraft*, Vol. 42, No. 4, 2005, pp. 991-997.

Murman, S.M., Aftosmis, M.J., and Berger, M.J., "Simulations of Store Separation from an F/A-18 with a Cartesian Method," *Journal of Aircraft*, Vol. 41, No. 4, 2004, pp. 870-878.

Niewoehner, R.J., and Filbey, J., "Computational Study of F/A-18 E/F Abrupt Wing Stall in the Approach Configuration," *Journal of Aircraft*, Vol. 42, No. 6, 2005, pp. 1516-1522.

Noel, P., Niewoehner, R., and Cenko, A., "Improved Understanding of CFD Predictions for Complex Aircraft/Store Configurations," AIAA Paper 4071, June 2003.

Pandya, M., et. al., "Recent Enhancements to USM3D Unstructured Flow Solver for Unsteady Flows," AIAA paper 2004-5201, August 2004.

Parikh, P., and Chung, J., "Computational Study of the Abrupt-Wing- Stall Characteristics of F/A-18E and F-16C," *Journal of Aircraft*, Vol. 42, No. 3, 2005, pp. 600–605.

Pirzadeh, S., and Frink, N., "Assessment of the Unstructured Grid Software TetrUSS for Drag Prediction of the DLR-F4 Configuration," AIAA Paper 2002-0839, January 2002.

Pirzadeh, S., "Unstructured Viscous Grid Generation by Advancing-Front Method," NASA Contractor Report 191449, 1993.

Rothback, N., and Cenko, A., "Dangers of Aircraft Modifications Without Conducting an Investigation into the Effects on the Aircraft Flowfield and Flying Qualities," 2001, pp. 1-14.

Sadeh, Y., and Gaton, V.A., "F-16/SPICE Separation Analysis-Summary of the Risk-Reduction Phase Program," *Journal of Aircraft*, Vol. 40, No. 1, 2003, pp. 70-76.

Woodson, S.H., Green, B.E., Chung, J.J, Grove, D.V., Parikh, P.C., and Forsythe, J.R., "Understanding Abrupt Wing Stall with Computational Fluid Dynamics," *Journal of Aircraft*, Vol. 42, No. 3, 2005, pp. 578-585.

APPENDIX A: CARRIAGE POSITION FORCE AND MOMENT DATA

Table 4. Force and moment coefficients on GBU-31 JDAM in carriage position with no pod attached

Mach number	Axial force coefficient, C_x	Sideforce coefficient, C_y	Normal force coefficient, C_z	Rolling moment coefficient, C_l	Pitching moment coefficient, C_m	Yawing moment coefficient, C_n
0.80	0.3746	-0.1370	0.2598	-0.0067	-2.1654	0.3070
0.85	0.5253	-0.1518	0.2835	-0.0103	-2.4308	0.3993
0.87	0.6040	-0.1599	0.3038	-0.0146	-2.5697	0.4679
0.88	0.6239	-0.1916	0.2828	-0.0221	-2.5126	0.6256
0.89	0.6637	-0.1927	0.1694	-0.0277	-2.0328	0.6801
0.90	0.7115	-0.1463	0.2298	-0.0191	-2.2978	0.4887
0.91	0.7369	-0.1375	0.3539	-0.0186	-2.8889	0.4717
0.92	0.7458	-0.1274	0.3563	-0.0179	-2.9169	0.4436
0.93	0.7477	-0.1183	0.3426	-0.0168	-2.8669	0.4150
0.94	0.7439	-0.1111	0.3299	-0.0155	-2.8200	0.3892
0.95	0.7382	-0.1056	0.3185	-0.0141	-2.7786	0.3669
0.98	0.7189	-0.0990	0.2875	-0.0103	-2.6548	0.3205
1.05	0.6878	-0.1210	0.2275	-0.0087	-2.3971	0.3452

Table 5. Force and moment coefficients on GBU-31 JDAM in carriage position with TFLIR pod attached

Mach number	Axial force coefficient, C_x	Sideforce coefficient, C_y	Normal force coefficient, C_z	Rolling moment coefficient, C_l	Pitching moment coefficient, C_m	Yawing moment coefficient, C_n
0.80	0.4417	-0.2843	0.3066	-0.0260	-2.4609	0.8542
0.85	0.5964	-0.2839	0.2899	-0.0229	-2.6223	0.8536
0.87	0.6735	-0.2974	0.2901	-0.0214	-2.7087	0.8935
0.88	0.7054	-0.2935	0.2582	-0.0178	-2.6287	0.8850
0.89	0.7274	-0.2835	0.1932	-0.0121	-2.4056	0.8431
0.90	0.7740	-0.3085	0.1093	-0.0169	-2.0569	0.9548
0.91	0.8214	-0.4223	0.1975	-0.0505	-2.4830	1.5149
0.92	0.8499	-0.4485	0.2491	-0.0570	-2.7593	1.6788
0.93	0.8638	-0.4608	0.2433	-0.0614	-2.7642	1.7773
0.94	0.8687	-0.4679	0.2211	-0.0671	-2.6873	1.8529
0.95	0.8702	-0.4699	0.2049	-0.0702	-2.6341	1.8989
0.98	0.8608	-0.4605	0.1750	-0.0735	-2.5446	1.9459
1.05	0.8231	-0.4327	0.1425	-0.0747	-2.4204	1.9392

Table 6. Force and moment coefficients on GBU-31 JDAM in carriage position with ATFLIR pod attached

Mach number	Axial force coefficient, C_x	Sideforce coefficient, C_y	Normal force coefficient, C_z	Rolling moment coefficient, C_l	Pitching moment coefficient, C_m	Yawing moment coefficient, C_n
0.80	0.4374	-0.3140	0.3189	-0.0244	-2.4639	0.8609
0.85	0.5873	-0.3139	0.3086	-0.0198	-2.6501	0.8417
0.87	0.6625	-0.3065	0.3050	-0.0131	-2.7277	0.7811
0.88	0.6972	-0.3159	0.2763	-0.0109	-2.6640	0.8158
0.89	0.7186	-0.2832	0.2118	-0.0018	-2.4342	0.6834
0.90	0.7559	-0.2704	0.1098	0.0027	-2.0110	0.6207
0.91	0.8075	-0.4318	0.0892	-0.0417	-1.9244	1.3790
0.92	0.8410	-0.5382	0.1964	-0.0820	-2.4400	1.9285
0.93	0.8606	-0.5402	0.2426	-0.0749	-2.6820	1.9868
0.94	0.8674	-0.5401	0.2346	-0.0780	-2.6680	2.0324
0.95	0.8699	-0.5352	0.2255	-0.0792	-2.6463	2.0493
0.98	0.8632	-0.5148	0.2039	-0.0791	-2.5889	2.0437
1.05	0.8271	-0.4795	0.1699	-0.0773	-2.4718	1.9988

Table 7. Force and moment coefficients on GBU-31 JDAM in carriage position with Litening pod attached

Mach number	Axial force coefficient, C_x	Sideforce coefficient, C_y	Normal force coefficient, C_z	Rolling moment coefficient, C_l	Pitching moment coefficient, C_m	Yawing moment coefficient, C_n
0.80	0.4903	-0.2951	0.2524	-0.0690	-2.5724	1.4210
0.85	0.6655	-0.2402	0.1670	-0.0641	-2.7364	1.3010
0.87	0.7390	-0.1629	0.1264	-0.0486	-2.6703	1.0310
0.88	0.7661	-0.1229	0.0803	-0.0391	-2.5181	0.8964
0.89	0.7969	-0.0485	-0.0013	-0.0221	-2.1862	0.5895
0.90	0.8190	-0.0445	-0.0389	-0.0121	-1.8219	0.4933
0.91	0.8666	-0.1481	0.0836	-0.0505	-2.6008	1.0704
0.92	0.8839	-0.2268	0.1292	-0.0707	-2.8346	1.4592
0.93	0.8949	-0.2833	0.1394	-0.0849	-2.9025	1.7512
0.94	0.9020	-0.3165	0.1314	-0.0961	-2.8832	1.9385
0.95	0.9086	-0.3278	0.1221	-0.1011	-2.8536	2.0224
0.98	0.8941	-0.3140	0.0990	-0.1024	-2.7736	2.0304
1.05	0.8518	-0.2846	0.0658	-0.1011	-2.6132	1.9766

APPENDIX B: CONVERGENCE ANALYSIS PROGRAM SCRIPT

```
function varargout = convtest(varargin)
% CONVTEST M-file for convtest.fig
%      CONVTEST, by itself, creates a new CONVTEST or raises the existing
%      singleton*.
%
%      H = CONVTEST returns the handle to a new CONVTEST or the handle to
%      the existing singleton*.
%
%      CONVTEST('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CONVTEST.M with the given input arguments.
%
%      CONVTEST('Property','Value',...) creates a new CONVTEST or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before convtest_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to convtest_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help convtest

% Last Modified by GUIDE v2.5 03-Sep-2007 23:01:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @convtest_OpeningFcn, ...
                   'gui_OutputFcn',  @convtest_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before convtest is made visible.
function convtest_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to convtest (see VARARGIN)

handles.dir = cd;
handles.data = [];
handles.plotcol = 1;
handles.inputfile = '';

set(handles.plotdatabutton,'enable','off');
set(handles.popbutton,'enable','off');

% Choose default command line output for convtest
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes convtest wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = convtest_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --------------------------------------------------------------------
function Filemenu_Callback(hObject, eventdata, handles)
% hObject    handle to Filemenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function selectfilebutton_Callback(hObject, eventdata, handles)
% hObject    handle to selectfilebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[fileinname, pathname] = uigetfile({'*.plt','USM3D Files (.plt)';'*.*','All Files (*.*)'},'Select hist.plt file');
if isequal(fileinname,0)
   disp('User selected Cancel')
   set(handles.plotdatabutton,'enable','off');
   set(handles.popbutton,'enable','off')
   set(handles.textfield,'String','No File Selected');
```

```
else
   set(handles.inputfile,'String',fileinname);

   cd(pathname);
   handles.data = dlmread(fileinname,'',7,0);

   filepath = ['Selected File: ',pathname,fileinname];
   set(handles.textfield,'String',filepath);
   cd(handles.dir);
   guidata(hObject, handles);

   set(handles.plotdatabutton,'enable','on');
   set(handles.popbutton,'enable','on')
end


% --------------------------------------------------------------------
function quitbutton_Callback(hObject, eventdata, handles)
% hObject    handle to quitbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(handles.figure1)

% --- Executes on selection change in plotmenu.
function plotmenu_Callback(hObject, eventdata, handles)
% hObject    handle to plotmenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% Hints: contents = get(hObject,'String') returns plotmenu contents as cell array
%        contents{get(hObject,'Value')} returns selected item from plotmenu


% --- Executes during object creation, after setting all properties.
function plotmenu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to plotmenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in plotdatabutton.
function plotdatabutton_Callback(hObject, eventdata, handles)
% hObject    handle to plotdatabutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
handles.plotcol = get(handles.plotmenu,'Value');
xvals = handles.data(:,1);
yvals = handles.data(:,handles.plotcol + 1);

lab = get(handles.plotmenu,'String');
plot(xvals,yvals),xlabel('Iteration number'),ylabel(lab(handles.plotcol))


% --- Executes on button press in popbutton.
function popbutton_Callback(hObject, eventdata, handles)
% hObject    handle to popbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.plotcol = get(handles.plotmenu,'Value');
xvals = handles.data(:,1);
yvals = handles.data(:,handles.plotcol + 1);

lab = get(handles.plotmenu,'String');
figure;plot(xvals,yvals),xlabel('Iteration number'),ylabel(lab(handles.plotcol))
```

APPENDIX C: UPWASH/SIDEWASH ANALYSIS PROGRAM SCRIPT

```
function varargout = dataanalysis(varargin)
% DATAANALYSIS M-file for dataanalysis.fig
%      DATAANALYSIS, by itself, creates a new DATAANALYSIS or raises the existing
%      singleton*.
%
%      H = DATAANALYSIS returns the handle to a new DATAANALYSIS or the handle to
%      the existing singleton*.
%
%      DATAANALYSIS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in DATAANALYSIS.M with the given input arguments.
%
%      DATAANALYSIS('Property','Value',...) creates a new DATAANALYSIS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before dataanalysis_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to dataanalysis_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dataanalysis

% Last Modified by GUIDE v2.5 10-Sep-2007 23:24:22

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dataanalysis_OpeningFcn, ...
                   'gui_OutputFcn',  @dataanalysis_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before dataanalysis is made visible.
```

```
function dataanalysis_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to dataanalysis (see VARARGIN)

handles.dir = cd;
handles.data = [];
handles.data2 = [];
handles.data3 = [];
handles.plotcol = 1;
handles.inputfile = '';
handles.inputfile2 = '';
handles.inputfile3 = '';
handles.alpha = [];
handles.beta = [];
handles.alpha2 = [];
handles.beta2 = [];
handles.alpha3 = [];
handles.beta3 = [];
handles.betam8bl88wl64 = dlmread('windtunnel','',[1 0 14 1]);
handles.betam95bl88wl64 = dlmread('windtunnel','',[1 2 18 3]);
handles.alpham8bl88wl64 = dlmread('windtunnel','',[1 4 18 5]);
handles.alpham95bl88wl64 = dlmread('windtunnel','',[1 6 16 7]);
handles.numfiles = 1;

set(handles.plotdatabutton,'enable','off');
set(handles.popbutton,'enable','off');

% Choose default command line output for dataanalysis
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dataanalysis wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = dataanalysis_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --------------------------------------------------------------------
function Filemenu_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to Filemenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function selectfilebutton_Callback(hObject, eventdata, handles)
% hObject    handle to selectfilebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cd('/home/Bill_CFD/cfd2/runsweng/datfiles');

[fileinname, pathname] = uigetfile({'*.dat','Tecplot Output files';'*.*','All Files (*.*)'},'Select Tecplot Output File');
if isequal(fileinname,0)
   disp('User selected Cancel')
   set(handles.plotdatabutton,'enable','off');
   set(handles.popbutton,'enable','off')
   set(handles.textfield,'String','No File Selected');
else
   set(handles.inputfile,'String',fileinname);

   cd(pathname);
   clear handles.data;
   handles.data = dlmread(fileinname,'',16,0);

   filepath = ['Selected File: ',pathname,fileinname];
   set(handles.textfield,'String',filepath);
   cd(handles.dir);

   handles.alpha = handles.data(:,6)./handles.data(:,4).*57.3;
   handles.beta = handles.data(:,5)./handles.data(:,4).*57.3;
   guidata(hObject, handles);

   set(handles.plotdatabutton,'enable','on');
   set(handles.popbutton,'enable','on')
end
cd(handles.dir);

% --------------------------------------------------------------------
function quitbutton_Callback(hObject, eventdata, handles)
% hObject    handle to quitbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(handles.figure1)

% --- Executes on selection change in plotmenu.
function plotmenu_Callback(hObject, eventdata, handles)
% hObject    handle to plotmenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: contents = get(hObject,'String') returns plotmenu contents as cell array
%        contents{get(hObject,'Value')} returns selected item from plotmenu


% --- Executes during object creation, after setting all properties.
function plotmenu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to plotmenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in plotdatabutton.
function plotdatabutton_Callback(hObject, eventdata, handles)
% hObject    handle to plotdatabutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



handles.plotcol = get(handles.plotmenu,'Value');
guidata(hObject, handles);


fl2 = 1;
fl3 = 1;
if strcmp(get(handles.fwdlimitbox,'String'),'default') == 0
   for a = 1:1000
      if handles.data(a,1) >= str2double(get(handles.fwdlimitbox,'String'))
         fl = a;
         break;
      end
   end
   if handles.numfiles > 1
   for a = 1:1000
      if handles.data2(a,1) >= str2double(get(handles.fwdlimitbox,'String'))
         fl2 = a;
         break;
      end
   end
   end
   if handles.numfiles > 2
   for a = 1:1000
      if handles.data3(a,1) >= str2double(get(handles.fwdlimitbox,'String'))
         fl3 = a;
         break;
      end
   end
```

```
    end
else
    fl = 1;
end


al = length(handles.data(:,1));
if handles.numfiles > 1
    al2 = length(handles.data2(:,1));
end
if handles.numfiles > 2
    al3 = length(handles.data3(:,1));
end
if strcmp(get(handles.aftlimitbox,'String'),'default') == 0
    for b = length(handles.data(:,1)):-1:1
        if handles.data(b,1) <= str2double(get(handles.aftlimitbox,'String'))
            al = b;
            break;
        end
    end
end
if handles.numfiles > 1 && strcmp(get(handles.aftlimitbox,'String'),'default') == 0
        al2 = length(handles.data2(:,1));
    for b = length(handles.data2(:,1)):-1:1
        if handles.data2(b,1) <= str2double(get(handles.aftlimitbox,'String'))
            al2 = b;
            break;
        end
    end
end
if handles.numfiles > 2 && strcmp(get(handles.aftlimitbox,'String'),'default') == 0
        al3 = length(handles.data3(:,1));
    for b = length(handles.data3(:,1)):-1:1
        if handles.data3(b,1) <= str2double(get(handles.aftlimitbox,'String'))
            al3 = b;
            break;
        end
    end
end




%%%%%%%%
xvals = handles.data(fl:al,1);
if handles.numfiles > 1
    xvals2 = handles.data2(fl2:al2,1);
end
if handles.numfiles > 2
    xvals3 = handles.data3(fl3:al3,1);
end

switch handles.numfiles
```

```
    case 1
        switch handles.plotcol
            case 1
                yvals = handles.alpha(fl:al);
            case 2
                yvals = handles.beta(fl:al);
        end
    case 2
        switch handles.plotcol
            case 1
                yvals = handles.alpha(fl:al);
                yvals2 = handles.alpha2(fl2:al2);
            case 2
                yvals = handles.beta(fl:al);
                yvals2 = handles.beta2(fl2:al2);
        end
    case 3
        switch handles.plotcol
            case 1
                yvals = handles.alpha(fl:al);
                yvals2 = handles.alpha2(fl2:al2);
                yvals3 = handles.alpha3(fl3:al3);
            case 2
                yvals = handles.beta(fl:al);
                yvals2 = handles.beta2(fl2:al2);
                yvals3 = handles.beta3(fl3:al3);
        end
end

lab = get(handles.plotmenu,'String');
switch handles.numfiles
    case 1
        sleg = handles.inputfile;
        plot(xvals,yvals,'b'),xlabel('Fuselage Station Number'),ylabel(lab(handles.plotcol)),legend(sleg)
        hold on;
    case 2
        sleg = {handles.inputfile, handles.inputfile2};
        plot(xvals,yvals,'b',xvals2,yvals2,'r'),xlabel('Fuselage Station Number'),ylabel(lab(handles.plotcol)), ...
            legend(sleg)
        hold on;
    case 3
        sleg = {handles.inputfile, handles.inputfile2, handles.inputfile3};
        plot(xvals,yvals,'b',xvals2,yvals2,'r',xvals3,yvals3,'g'),xlabel('Fuselage Station Number'),...
            ylabel(lab(handles.plotcol)), legend(sleg)
        hold on;
end

switch get(handles.wtmenu,'Value')
    case 1                      %No WT data
        hold off;
    case 2                      %WT M8
        switch handles.plotcol
            case 1              %Alpha
```

```
            wtxvals = handles.alpham8bl88wl64(:,1);
            wtyvals = handles.alpham8bl88wl64(:,2);
            plot(wtxvals,wtyvals,'ks-'),legend(sleg,'WT M.8 BL88 WL64')
            hold off;
        case 2                      %Beta
            wtxvals = handles.betam8bl88wl64(:,1);
            wtyvals = handles.betam8bl88wl64(:,2);
            plot(wtxvals,wtyvals,'ks-'),legend(sleg,'WT M.8 BL88 WL64')
            hold off;
    end
  case 3                            %WT M9
    switch handles.plotcol
        case 1
            wtxvals = handles.alpham95bl88wl64(:,1);
            wtyvals = handles.alpham95bl88wl64(:,2);
            plot(wtxvals,wtyvals,'ks-'),legend(sleg,'WT M.95 BL88 WL64')
            hold off;
        case 2
            wtxvals = handles.betam95bl88wl64(:,1);
            wtyvals = handles.betam95bl88wl64(:,2);
            plot(wtxvals,wtyvals,'ks-'),legend(sleg,'WT M.95 BL88 WL64')
            hold off;
    end
end


% --- Executes on button press in popbutton.
function popbutton_Callback(hObject, eventdata, handles)
% hObject    handle to popbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.plotcol = get(handles.plotmenu,'Value');
guidata(hObject, handles);


fl2 = 1;
fl3 = 1;
if strcmp(get(handles.fwdlimitbox,'String'),'default') == 0
  for a = 1:1000
    if handles.data(a,1) >= str2double(get(handles.fwdlimitbox,'String'))
        fl = a;
        break;
    end
  end
  if handles.numfiles > 1
  for a = 1:1000
    if handles.data2(a,1) >= str2double(get(handles.fwdlimitbox,'String'))
        fl2 = a;
        break;
    end
  end
  end
```

```
    if handles.numfiles > 2
    for a = 1:1000
        if handles.data3(a,1) >= str2double(get(handles.fwdlimitbox,'String'))
            fl3 = a;
            break;
        end
    end
    end
else
    fl = 1;
end


al = length(handles.data(:,1));
if handles.numfiles > 1
    al2 = length(handles.data2(:,1));
end
if handles.numfiles > 2
    al3 = length(handles.data3(:,1));
end
if strcmp(get(handles.aftlimitbox,'String'),'default') == 0
    for b = length(handles.data(:,1)):-1:1
        if handles.data(b,1) <= str2double(get(handles.aftlimitbox,'String'))
            al = b;
            break;
        end
    end
end
if handles.numfiles > 1 && strcmp(get(handles.aftlimitbox,'String'),'default') == 0
        al2 = length(handles.data2(:,1));
    for b = length(handles.data2(:,1)):-1:1
        if handles.data2(b,1) <= str2double(get(handles.aftlimitbox,'String'))
            al2 = b;
            break;
        end
    end
end
if handles.numfiles > 2 && strcmp(get(handles.aftlimitbox,'String'),'default') == 0
        al3 = length(handles.data3(:,1));
    for b = length(handles.data3(:,1)):-1:1
        if handles.data3(b,1) <= str2double(get(handles.aftlimitbox,'String'))
            al3 = b;
            break;
        end
    end
end



%%%%%%%%
xvals = handles.data(fl:al,1);
if handles.numfiles > 1
```

```
    xvals2 = handles.data2(fl2:al2,1);
end
if handles.numfiles > 2
    xvals3 = handles.data3(fl3:al3,1);
end

switch handles.numfiles
    case 1
        switch handles.plotcol
            case 1
                yvals = handles.alpha(fl:al);
            case 2
                yvals = handles.beta(fl:al);
        end
    case 2
        switch handles.plotcol
            case 1
                yvals = handles.alpha(fl:al);
                yvals2 = handles.alpha2(fl2:al2);
            case 2
                yvals = handles.beta(fl:al);
                yvals2 = handles.beta2(fl2:al2);
        end
    case 3
        switch handles.plotcol
            case 1
                yvals = handles.alpha(fl:al);
                yvals2 = handles.alpha2(fl2:al2);
                yvals3 = handles.alpha3(fl3:al3);
            case 2
                yvals = handles.beta(fl:al);
                yvals2 = handles.beta2(fl2:al2);
                yvals3 = handles.beta3(fl3:al3);
        end
end

lab = get(handles.plotmenu,'String');
switch handles.numfiles
    case 1
        sleg = handles.inputfile;
        figure(2);plot(xvals,yvals,'b'),xlabel('Fuselage Station Number'),ylabel(lab(handles.plotcol)),legend(sleg)
        hold on;
    case 2
        sleg = {handles.inputfile, handles.inputfile2};
        figure(2);plot(xvals,yvals,'b',xvals2,yvals2,'r'),xlabel('Fuselage Station Number'),ylabel(lab(handles.plotcol)), ...
            legend(sleg)
        hold on;
    case 3
        sleg = {handles.inputfile, handles.inputfile2, handles.inputfile3};
        figure(2);plot(xvals,yvals,'b',xvals2,yvals2,'r',xvals3,yvals3,'g'),xlabel('Fuselage Station Number'),...
            ylabel(lab(handles.plotcol)), legend(sleg)
        hold on;
end
```

```
switch get(handles.wtmenu,'Value')
   case 1                        %No WT data
      hold off;
   case 2                        %WT M8
      switch handles.plotcol
         case 1                  %Alpha
            wtxvals = handles.alpham8bl88wl64(:,1);
            wtyvals = handles.alpham8bl88wl64(:,2);
            plot(wtxvals,wtyvals,'ks-'),legend(sleg,'WT M.8 BL88 WL64')
            hold off;
         case 2                  %Beta
            wtxvals = handles.betam8bl88wl64(:,1);
            wtyvals = handles.betam8bl88wl64(:,2);
            plot(wtxvals,wtyvals,'ks-'),legend(sleg,'WT M.8 BL88 WL64')
            hold off;
      end
   case 3                        %WT M9
      switch handles.plotcol
         case 1
            wtxvals = handles.alpham95bl88wl64(:,1);
            wtyvals = handles.alpham95bl88wl64(:,2);
            plot(wtxvals,wtyvals,'ks-'),legend(sleg,'WT M.95 BL88 WL64')
            hold off;
         case 2
            wtxvals = handles.betam95bl88wl64(:,1);
            wtyvals = handles.betam95bl88wl64(:,2);
            plot(wtxvals,wtyvals,'ks-'),legend(sleg,'WT M.95 BL88 WL64')
            hold off;
      end
end



function fwdlimitbox_Callback(hObject, eventdata, handles)
% hObject    handle to fwdlimitbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fwdlimitbox as text
%        str2double(get(hObject,'String')) returns contents of fwdlimitbox as a double


% --- Executes during object creation, after setting all properties.
function fwdlimitbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fwdlimitbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
```

```
end


function aftlimitbox_Callback(hObject, eventdata, handles)
% hObject    handle to aftlimitbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of aftlimitbox as text
%        str2double(get(hObject,'String')) returns contents of aftlimitbox as a double


% --- Executes during object creation, after setting all properties.
function aftlimitbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to aftlimitbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in wtmenu.
function wtmenu_Callback(hObject, eventdata, handles)
% hObject    handle to wtmenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns wtmenu contents as cell array
%        contents{get(hObject,'Value')} returns selected item from wtmenu


% --- Executes during object creation, after setting all properties.
function wtmenu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to wtmenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in numfilemenu.
function numfilemenu_Callback(hObject, eventdata, handles)
% hObject    handle to numfilemenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns numfilemenu contents as cell array
%        contents{get(hObject,'Value')} returns selected item from numfilemenu


% --- Executes during object creation, after setting all properties.
function numfilemenu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numfilemenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in loadfilebutton.
function loadfilebutton_Callback(hObject, eventdata, handles)
% hObject    handle to loadfilebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cd('/home/Bill_CFD/cfd2');
clear handles.data handles.data2 handles.data3
guidata(hObject, handles);

[fileinname, pathname] = uigetfile({'*.dat','Tecplot Output files';'*.*','All Files (*.*)'},'Select First Tecplot Output
File');
if isequal(fileinname,0)
   disp('User selected Cancel')
   set(handles.plotdatabutton,'enable','off');
   set(handles.popbutton,'enable','off')
   set(handles.textfield,'String','No File Selected');
else
   %set(handles.inputfile,'String',fileinname);
   handles.inputfile = fileinname;
   cd(pathname);
   clear handles.data;
   handles.data = dlmread(fileinname,'',16,0);

   filepath = ['First File: ',pathname,fileinname];
   set(handles.textfield,'String',filepath);
   cd(handles.dir);

   handles.alpha = handles.data(:,6)./handles.data(:,4).*57.3;
   handles.beta = handles.data(:,5)./handles.data(:,4).*57.3;
   guidata(hObject, handles);

   set(handles.plotdatabutton,'enable','on');
   set(handles.popbutton,'enable','on')
```

```
end

if get(handles.numfilemenu,'Value') > 1
   cd('/home/Bill_CFD/cfd2');
   [fileinname2, pathname2] = uigetfile({'*.dat','Tecplot Output files';'*.*','All Files (*.*)'},'Select Second Tecplot
Output File');
   if isequal(fileinname2,0)
      disp('User selected Cancel')
      set(handles.textfield2,'String','No File Selected');
   else
      handles.inputfile2 = fileinname2;
      cd(pathname2);
      clear handles.data2;
      handles.data2 = dlmread(fileinname2,'',16,0);

      filepath2 = ['Second File: ',pathname2,fileinname2];
      set(handles.textfield2,'String',filepath2);
      cd(handles.dir);

      handles.alpha2 = handles.data2(:,6)./handles.data2(:,4).*57.3;
      handles.beta2 = handles.data2(:,5)./handles.data2(:,4).*57.3;
      guidata(hObject, handles);
   end
end

if get(handles.numfilemenu,'Value') > 2
   cd('/home/Bill_CFD/cfd2');
   [fileinname3, pathname3] = uigetfile({'*.dat','Tecplot Output files';'*.*','All Files (*.*)'},'Select Third Tecplot
Output File');
   if isequal(fileinname3,0)
      disp('User selected Cancel')
      set(handles.textfield3,'String','No File Selected');
   else
      handles.inputfile3 = fileinname3;
      cd(pathname3);
      clear handles.data3;
      handles.data3 = dlmread(fileinname3,'',16,0);

      filepath3 = ['Third File: ',pathname3,fileinname3];
      set(handles.textfield3,'String',filepath3);
      cd(handles.dir);

      handles.alpha3 = handles.data3(:,6)./handles.data3(:,4).*57.3;
      handles.beta3 = handles.data3(:,5)./handles.data3(:,4).*57.3;
      guidata(hObject, handles);
   end
end

if length(handles.data2) > 0
   handles.numfiles = 2;
end
if length(handles.data3) > 0
   handles.numfiles = 3;
```

```
end
guidata(hObject, handles);

cd(handles.dir);
```